US006665317B1

(12) **United States Patent**
Scott

(10) **Patent No.: US 6,665,317 B1**
(45) **Date of Patent: Dec. 16, 2003**

(54) **METHOD, SYSTEM, AND COMPUTER PROGRAM PRODUCT FOR MANAGING JITTER**

(75) Inventor: **Mark Scott**, Ashburn, VA (US)

(73) Assignee: **Array Telecom Corporation**, Herndon, VA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/429,652**

(22) Filed: **Oct. 29, 1999**

(51) **Int. Cl.**[7] **H04J 3/06**

(52) **U.S. Cl.** **370/516**; 370/253; 370/353; 375/371

(58) **Field of Search** 370/252–256, 370/401, 516–518, 352, 353, 354, 355, 356, 465, 466, 503; 375/354, 371

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,894,823	A *	1/1990	Adelmann et al.	370/252
5,467,342	A *	11/1995	Logston et al.	370/253
5,621,727	A	4/1997	Vaudreuil	370/60
5,652,627	A *	7/1997	Allen	348/497
5,742,596	A	4/1998	Baratz et al.	370/356
5,757,871	A	5/1998	Furukawa et al.	375/372
5,790,538	A	8/1998	Sugar	370/352
5,790,543	A	8/1998	Cloutier	370/395
5,805,602	A *	9/1998	Cloutier et al.	370/516
5,812,840	A	9/1998	Shwartz	395/604
5,870,464	A	2/1999	Brewster et al.	379/219

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

WO	WO 97/14238 A1	4/1997	H04L/12/46
WO	WO 97/23078 A1	6/1997	H04L/12/56
WO	WO 97/27692 A1	7/1997	H04L/12/56
WO	WO 97/28628 A1	8/1997	H04L/12/56

OTHER PUBLICATIONS

Pogrebinsky et al. (US PUB US2002/0101885) discloses jitter buffer and methods for control of same.*

Prosise, Jeff, "Programming Windows 95 with MFC, Part VII: The Document/View Architecture," Microsoft Systems Journal (Redmond, Washington), 35 pages, Feb. 1996.

Anquetil, L–P et al., "Media Gateway Control Protocol and Voice Over IP Gateways: MGCP and VoIP Gateways Will Offer Seamless Interworking of New VoIP Networks with Today's Telephone Networks," Alcatel Communications Review–2nd Quarter, Alcatel Corporate Research Center, Marcoussis, France, Apr. 1999, pp. 151–157.

Copy of International Search Report, issued Aug. 29, 2000, for PCT/US00/02330, 5 pages.

Array Series 3000 Users Manual, Array Telecom Corp, Entire Manual (Aug. 27, 1999).

ctvoice IP Telephony, Product Brochure, Comdial, 6 Pages (Copy obtained Aug. 1999).

ctvoice System User's Manual, Comdial, Entire Manual (Jun. 1998).

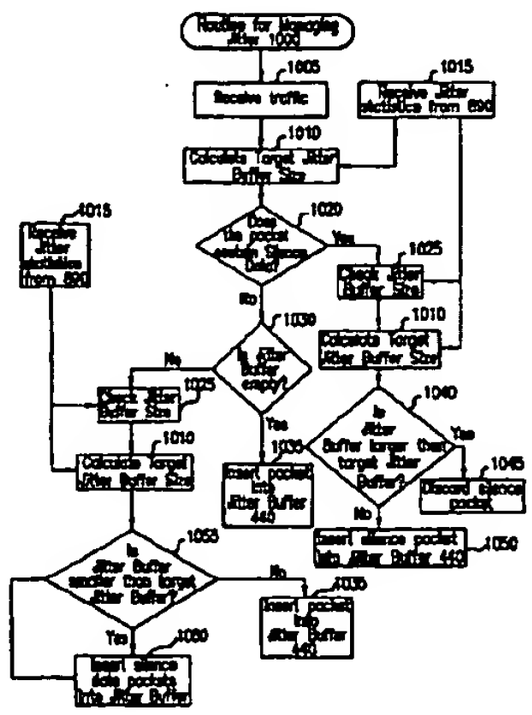
Yang, C., *INETPhone: Telephone Services and Servers on Internet*, at http://www.ds.internic.net/rfc/rfc1789.txt, 6 pages, (Apr. 1995).

Primary Examiner—Ricky Ngo
Assistant Examiner—Yvonne Q. Ha
(74) *Attorney, Agent, or Firm*—Sterne, Kessler, Goldstein & Fox, P.L.L.C.

(57) **ABSTRACT**

A method, system and computer software product for managing jitter buffering that accurately measures network latency, the variation in latency (also known as jitter), and efficiently manages the media packet stream and jitter buffers is disclosed. A framer time-stamps incoming packets. A traffic analyzer maintains a sliding window of statistics generated from a recent set of packets. A jitter manager monitors packets, receives and makes adjustments based on information received from the traffic analyzer, and manages any connected jitter buffers.

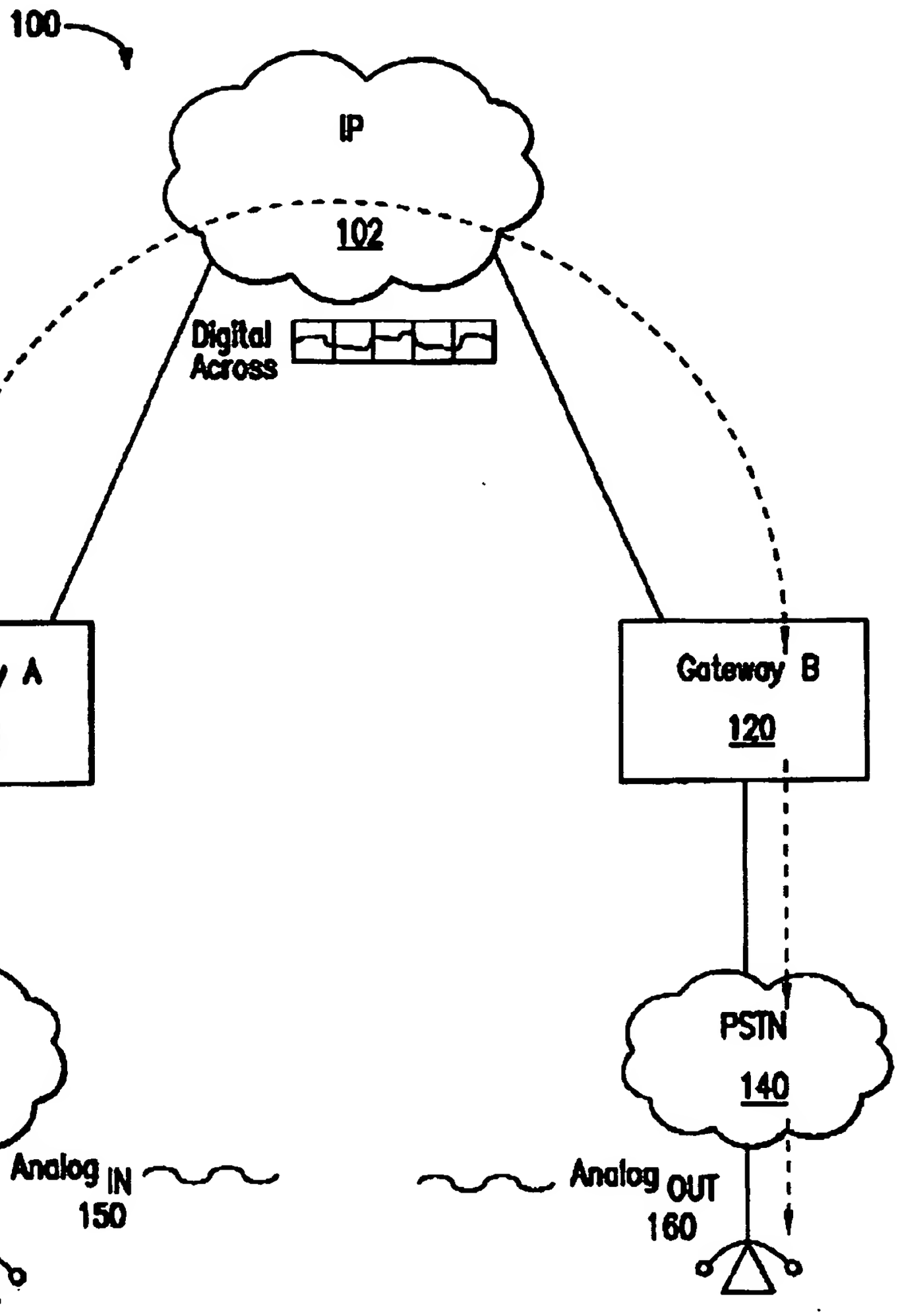
19 Claims, 14 Drawing Sheets



U.S. PATENT DOCUMENTS

5,892,822	A	4/1999	Gottlieb et al.	379/220
5,897,613	A	4/1999	Chan	704/210
5,900,000	A	5/1999	Korenshtein	707/200
5,940,479	A	8/1999	Guy et al.	379/93.01
5,940,827	A	8/1999	Hapner et al.	707/8
5,940,829	A	8/1999	Tsuiki et al.	707/10
5,940,832	A	8/1999	Hamada et al.	707/100
5,953,405	A	9/1999	Miloslavsky	379/265
5,966,387	A *	10/1999	Cloutier	370/516
5,991,307	A *	11/1999	Komuro et al.	370/473
6,011,899	A *	1/2000	Ohishi et al.	386/98
6,259,677	B1 *	7/2001	Jain	370/252
6,389,032	B1 *	5/2002	Cohen	370/412
6,452,915	B1 *	9/2002	Jorgensen	370/338
6,452,950	B1 *	9/2002	Ohlsson et al.	370/516

* cited by examiner



Voice Over Internet Protocol Architecture

FIG.1

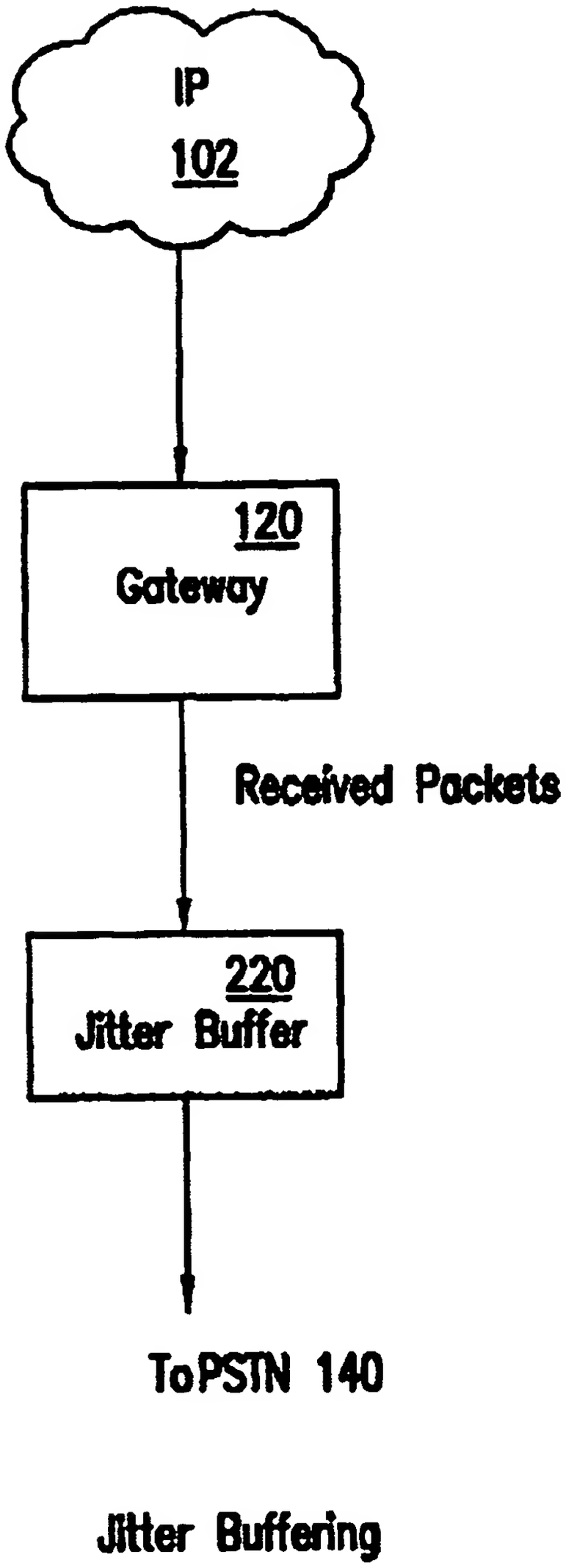

IP
102
120
Gateway
Received Packets
220
Jitter Buffer
ToPSTN 140
Jitter Buffering

FIG.2

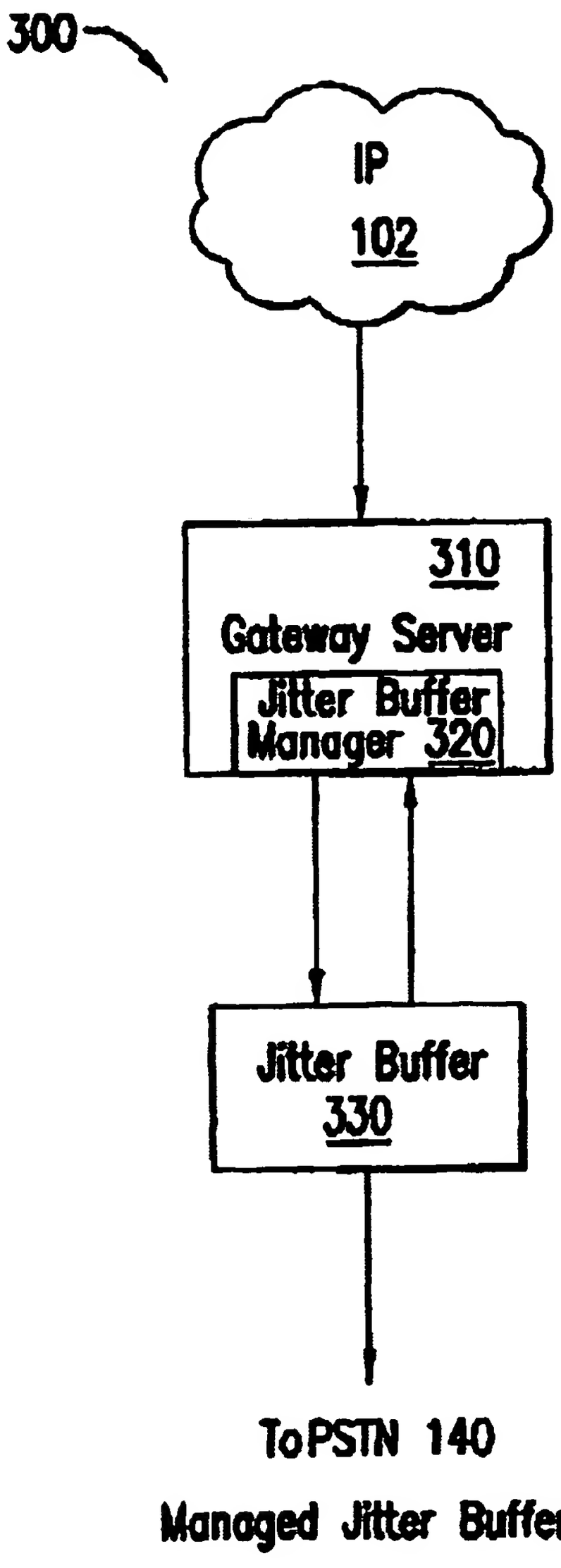

300
IP
102
310
Gateway Server
Jitter Buffer Manager 320
Jitter Buffer 330
ToPSTN 140
Managed Jitter Buffering

FIG.3

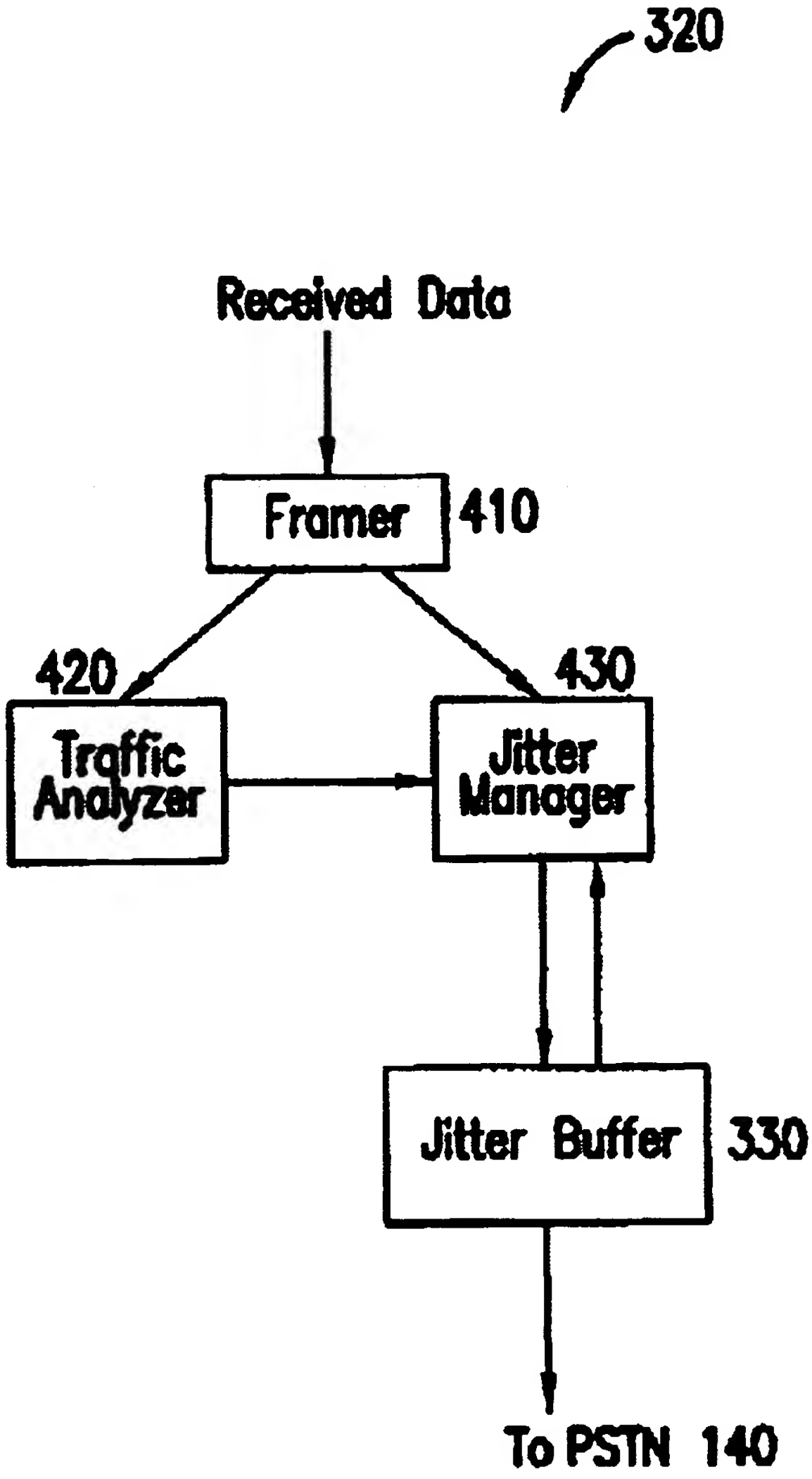

320
Received Data
Framer
410
420
430
Traffic Analyzer
Jitter Manager
Jitter Buffer
330
To PSTN 140

FIG.4

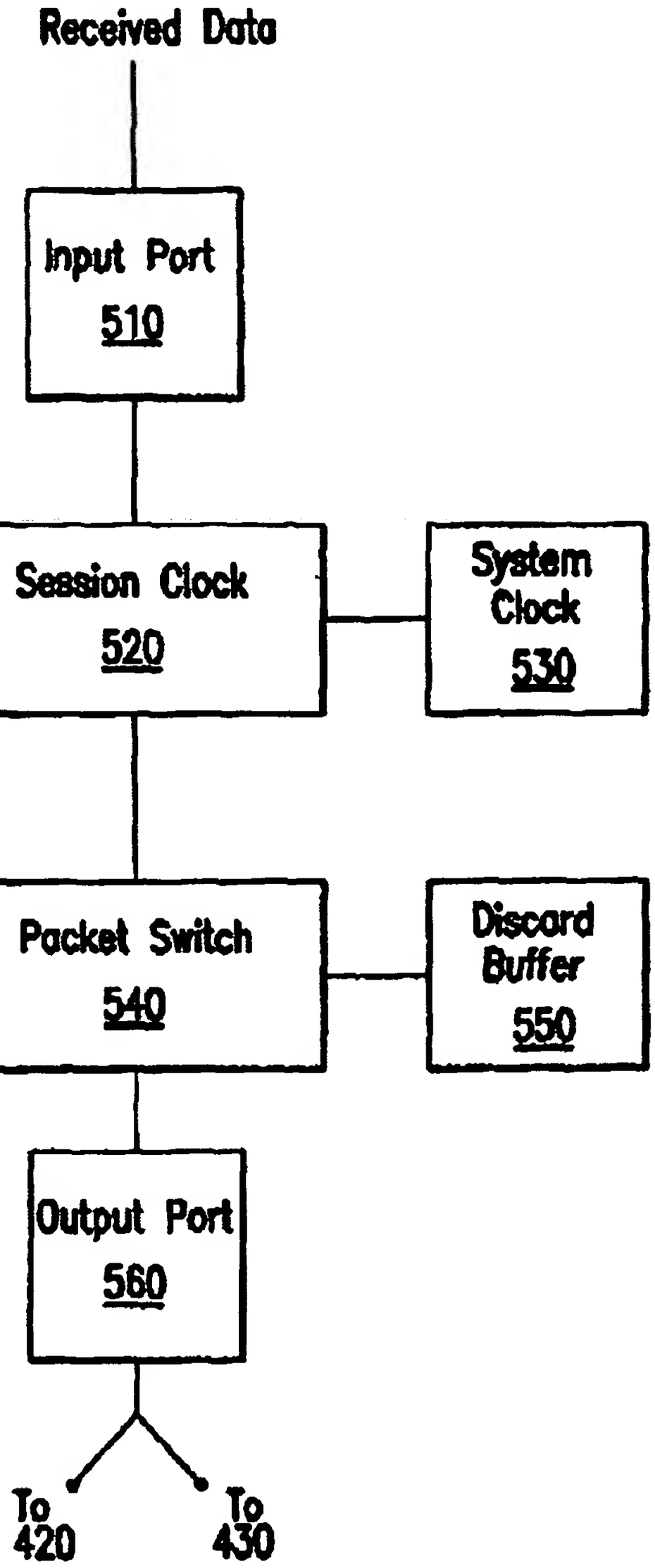

Received Data
Input Port
510
Session Clock
520
System Clock
530
Packet Switch
540
Discard Buffer
550
Output Port
560
To 420
To 430

FIG.5

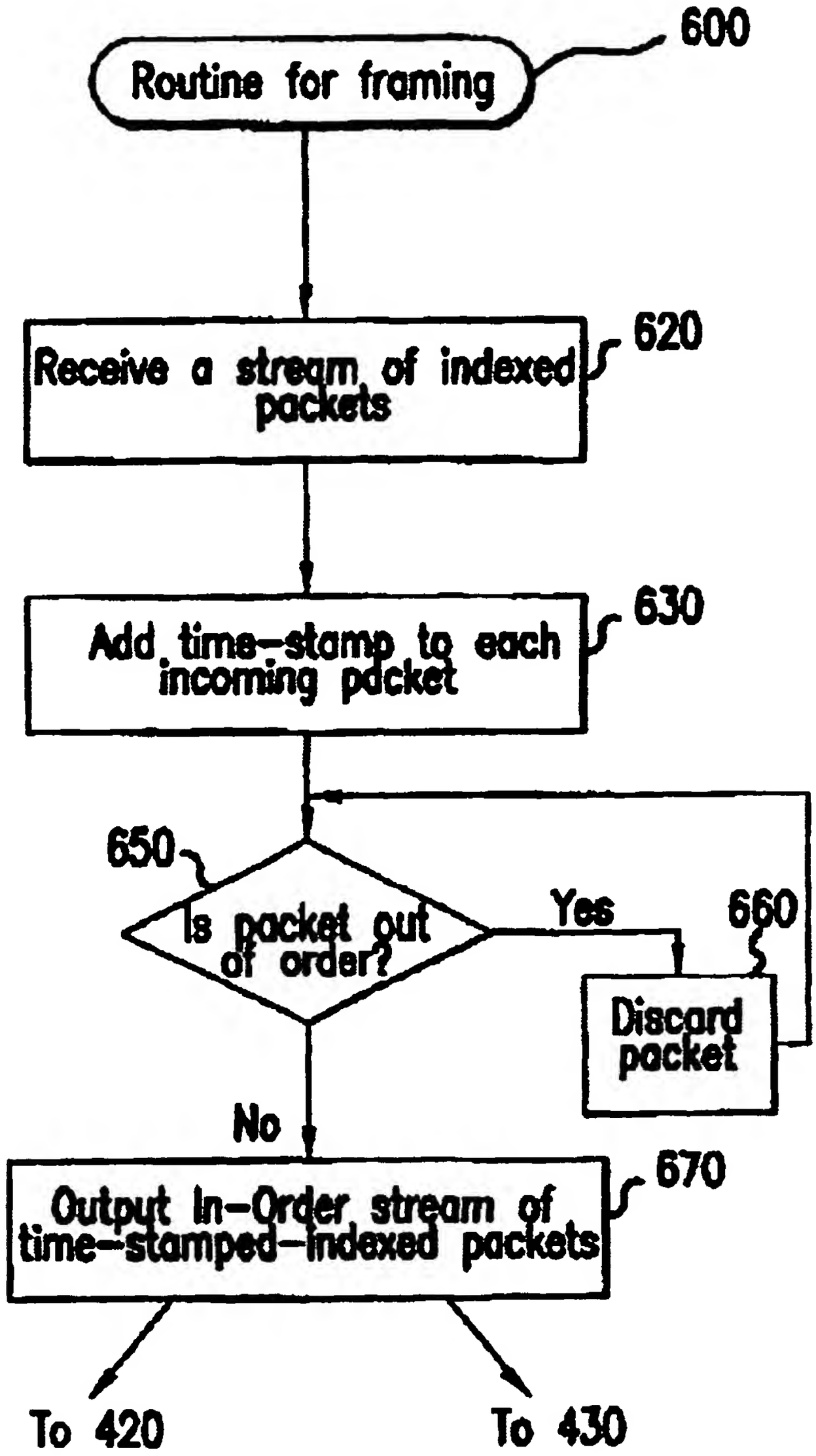

600
Routine for framing
620
Receive a stream of indexed packets
630
Add time-stamp to each incoming packet
650
Is packet out of order?
Yes
660
Discard packet
No
670
Output In-Order stream of time-stamped-indexed packets
To 420
To 430

FIG.6

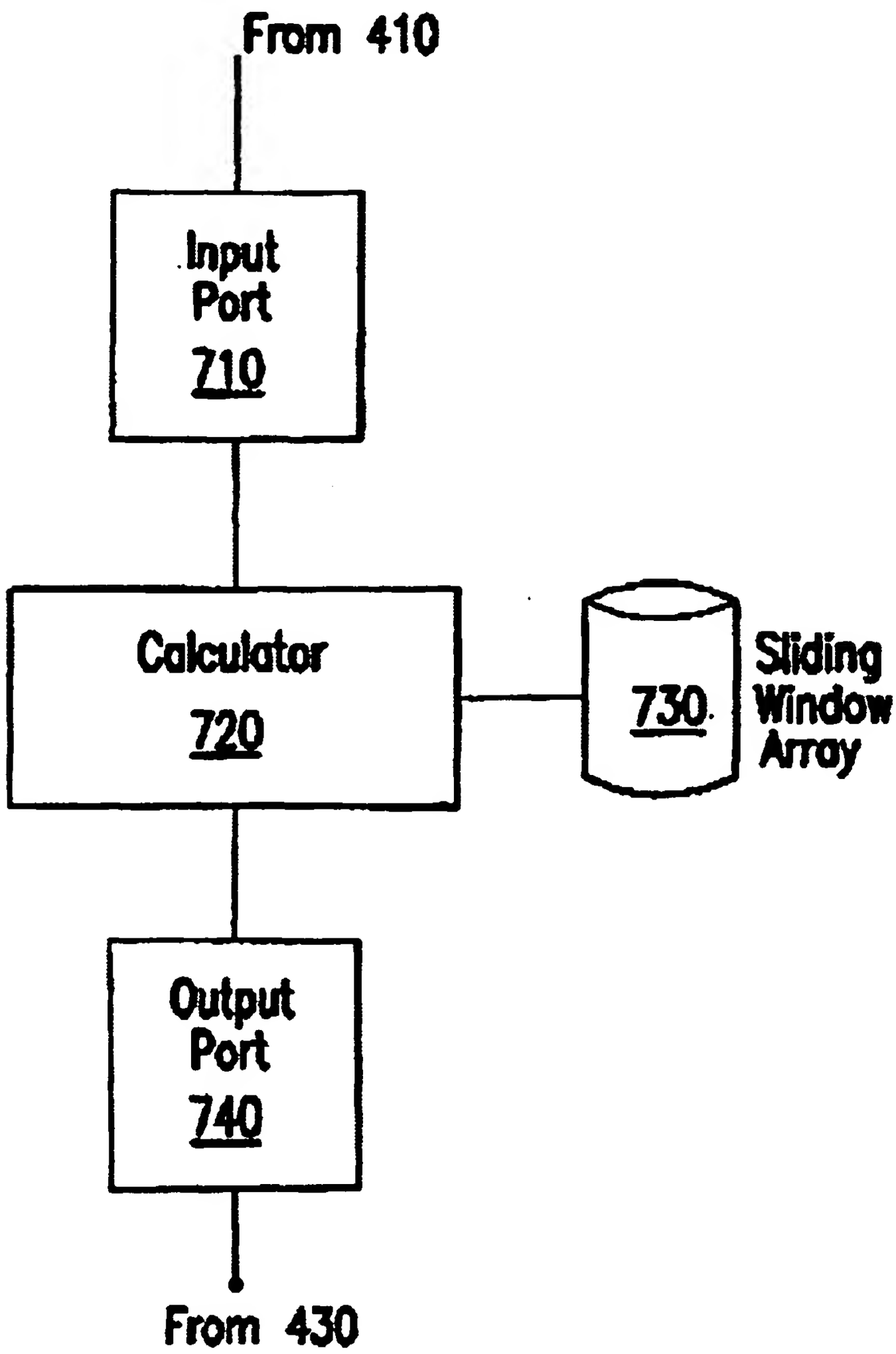

From 410
Input Port 710
Calculator 720
730
Sliding Window Array
Output Port 740
From 430

FIG.7

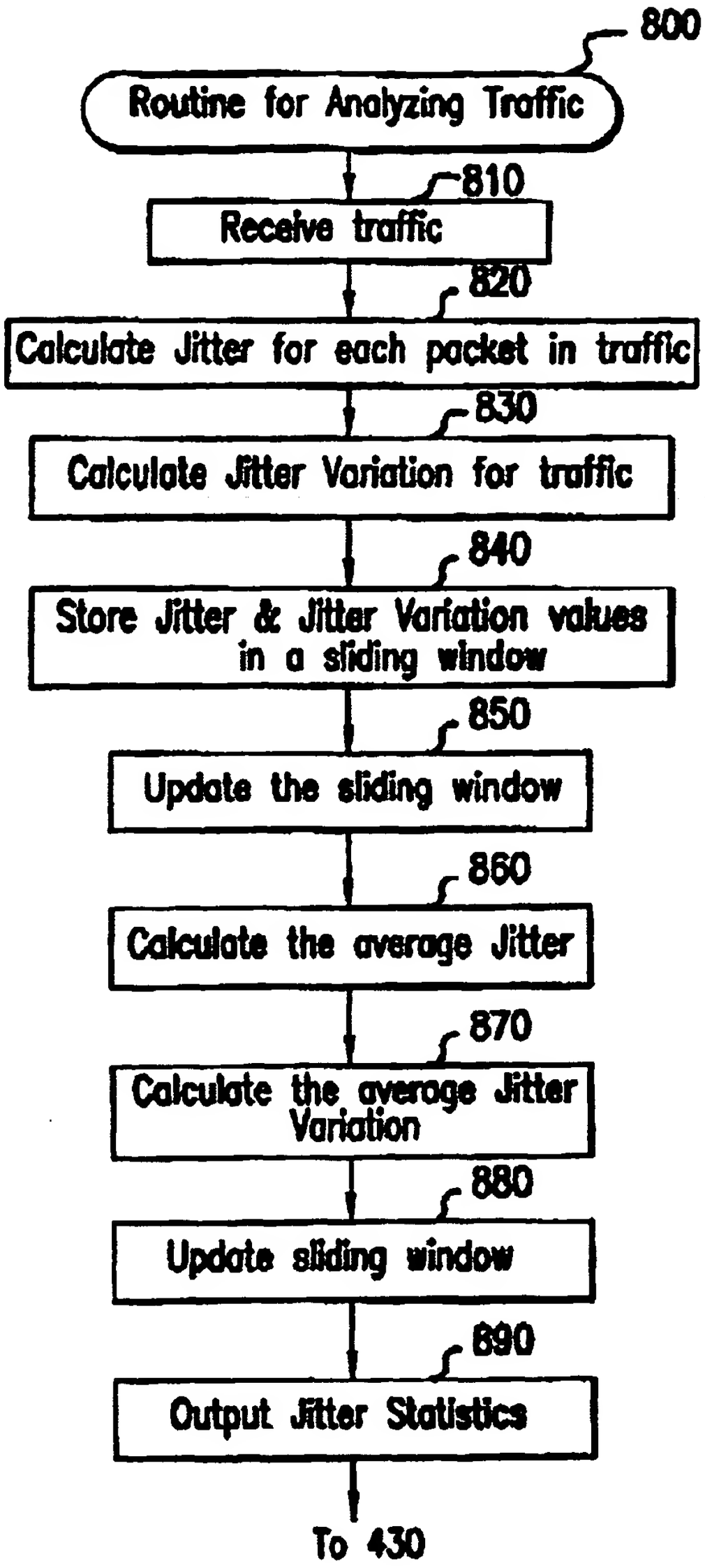

800
Routine for Analyzing Traffic
810
Receive traffic
820
Calculate Jitter for each packet in traffic
830
Calculate Jitter Variation for traffic
840
Store Jitter & Jitter Variation values in a sliding window
850
Update the sliding window
860
Calculate the average Jitter
870
Calculate the average Jitter Variation
880
Update sliding window
890
Output Jitter Statistics
To 430

FIG.8

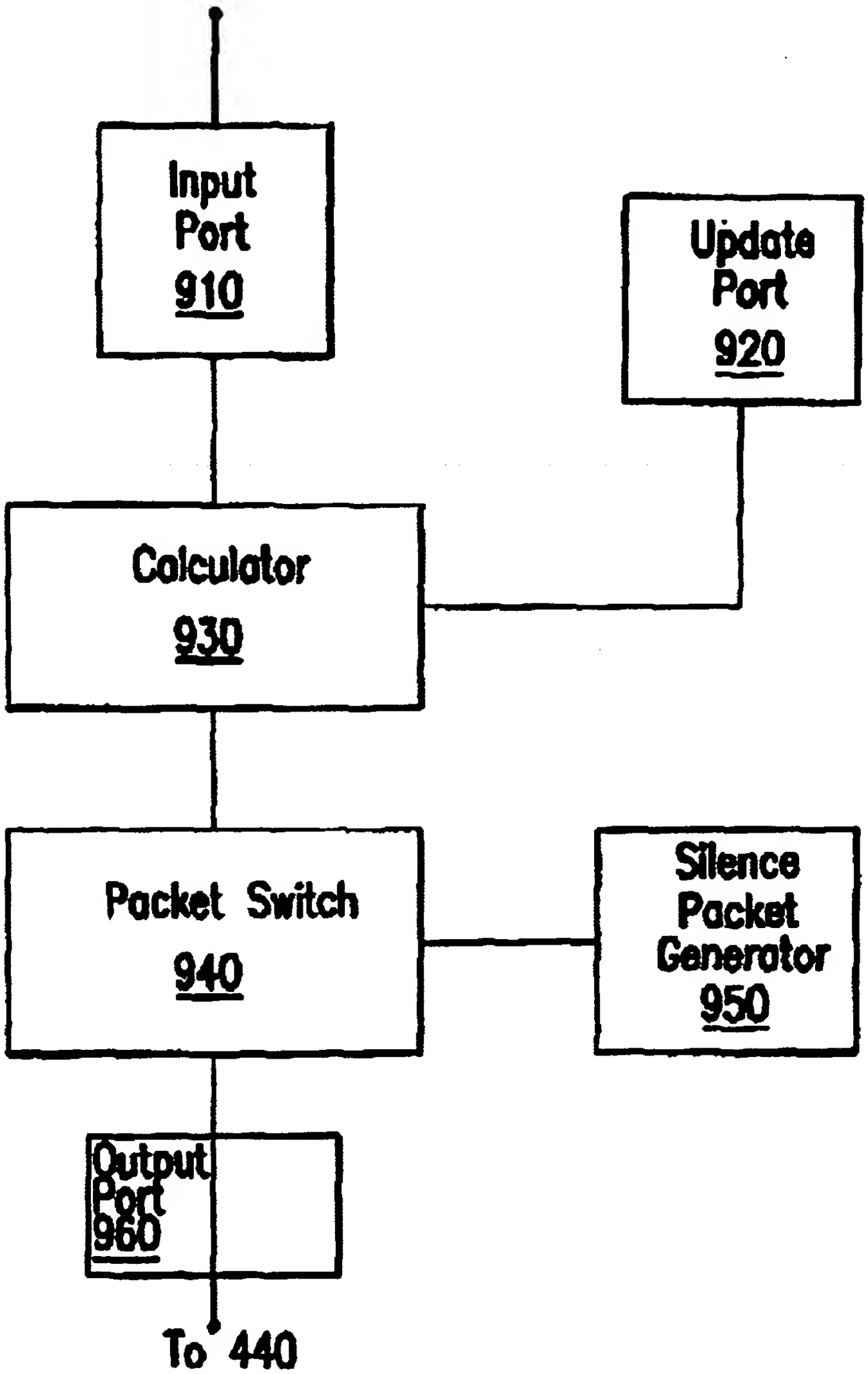

Input Port 910
Update Port 920
Calculator 930
Packet Switch 940
Silence Packet Generator 950
Output Port 960
To 440

FIG.9

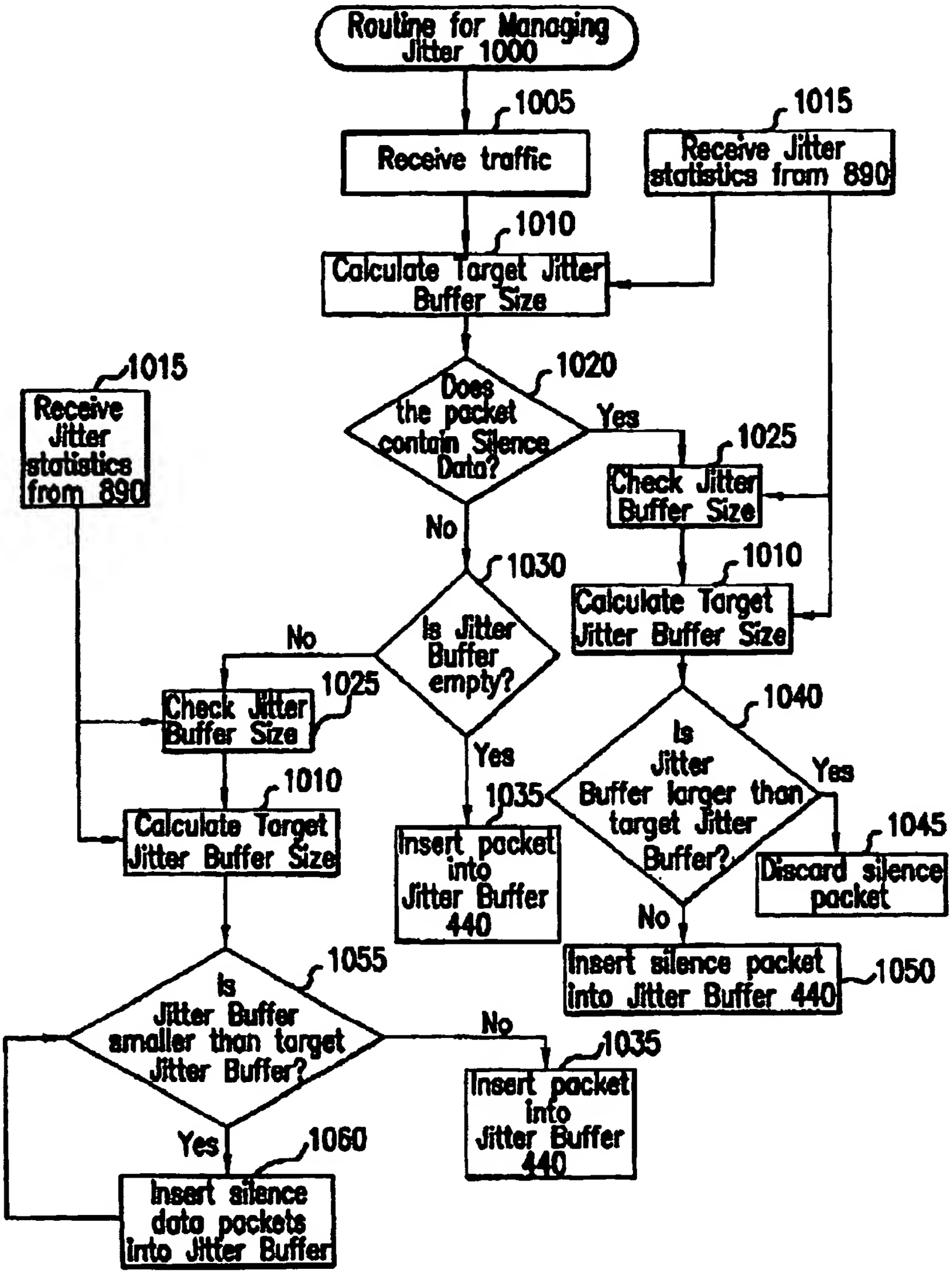

Routine for Managing Jitter 1000
1005
Receive traffic
1010
Calculate Target Jitter Buffer Size
1015
Receive Jitter statistics from 890
1020
Does the packet contain Silence Data?
Yes
1025
Check Jitter Buffer Size
No
1030
Is Jitter Buffer empty?
No
1025
Check Jitter Buffer Size
1010
Calculate Target Jitter Buffer Size
Yes
1035
Insert packet into Jitter Buffer 440
1040
Is Jitter Buffer larger than target Jitter Buffer?
Yes
1045
Discard silence packet
No
Insert silence packet into Jitter Buffer 440
1050
1015
Receive Jitter statistics from 890
1055
Is Jitter Buffer smaller than target Jitter Buffer?
No
1035
Insert packet into Jitter Buffer 440
Yes
1060
Insert silence data packets into Jitter Buffer

FIG.10

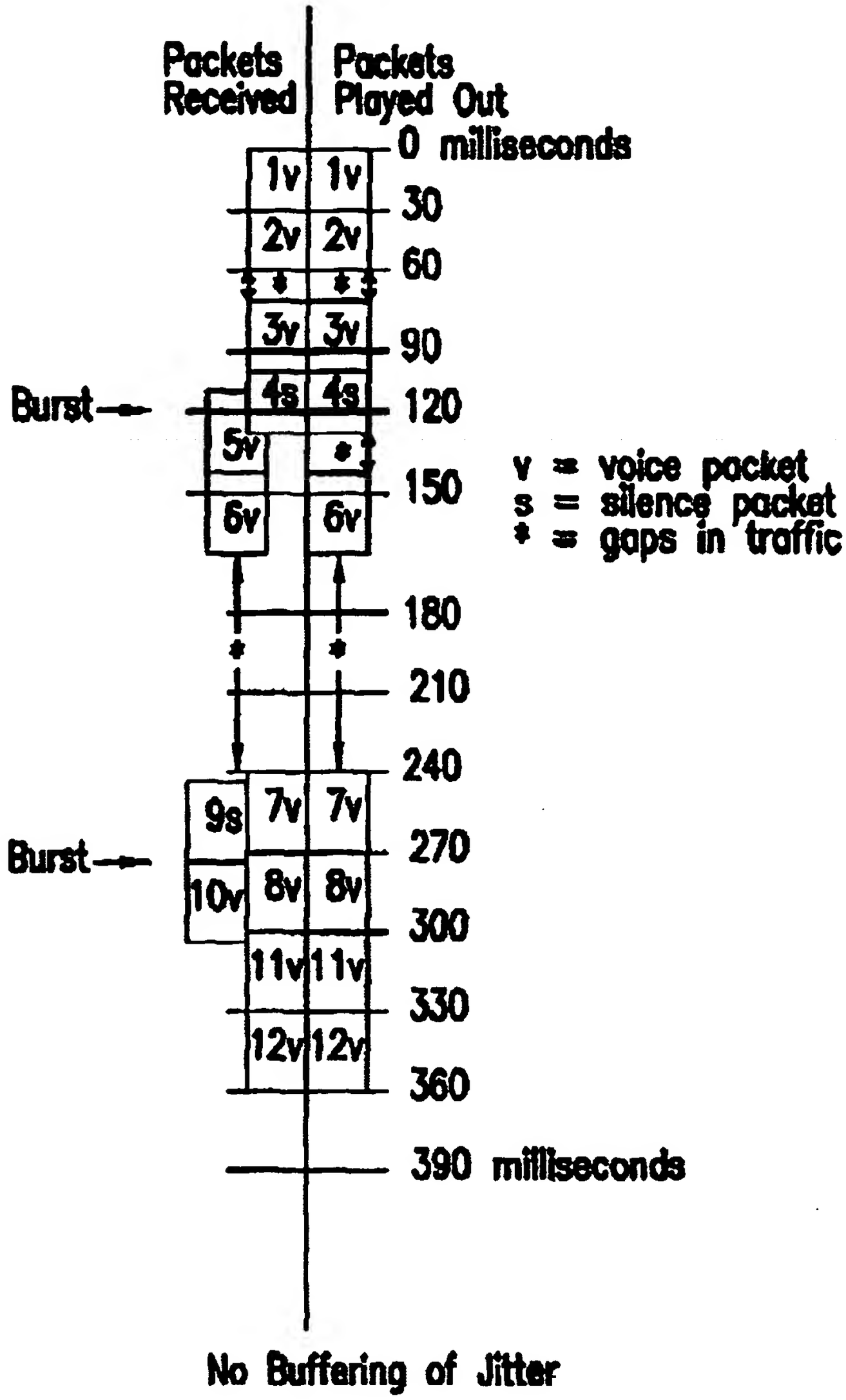
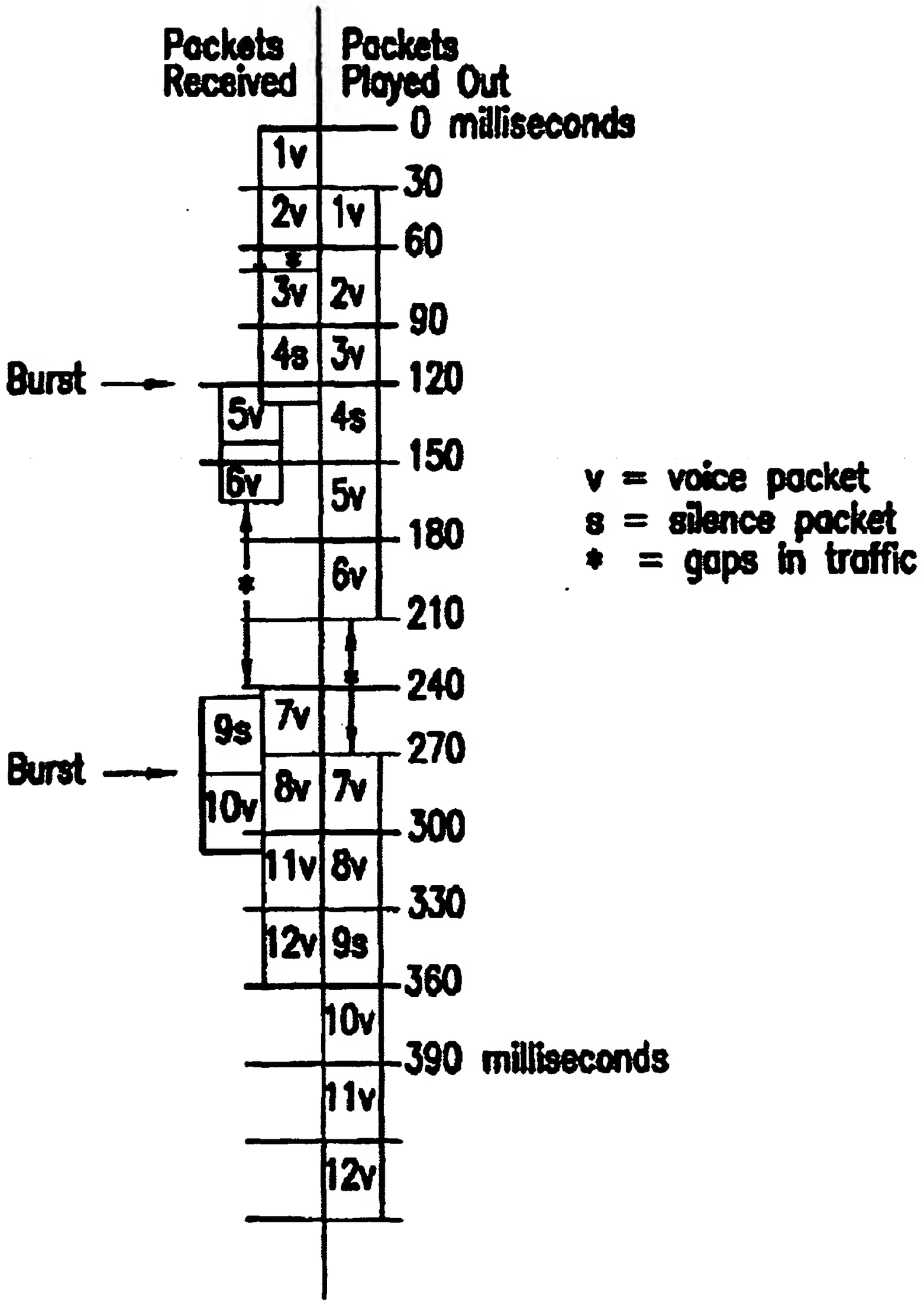
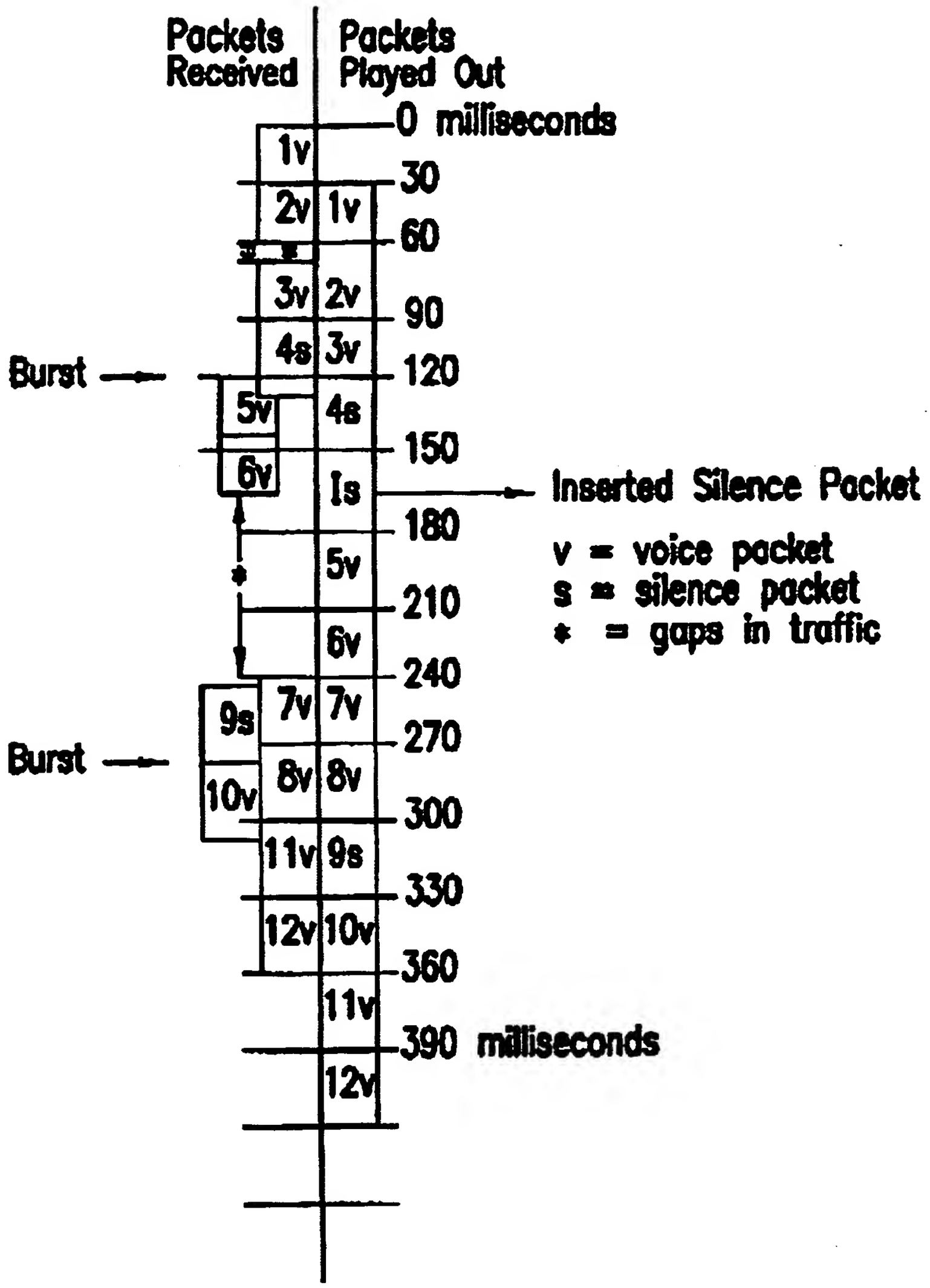

Packets Received
Packets Played Out
0 milliseconds
1v 1v
30
2v 2v
60
3v 3v
90
Burst
4s 4s
120
5v *
150
6v 6v
180
210
240
9s 7v 7v
Burst
270
10v 8v 8v
300
11v 11v
330
12v 12v
360
390 milliseconds
v = voice packet
s = silence packet
* = gaps in traffic
No Buffering of Jitter

FIG. 11



Static Buffering of Jitter

FIG. 12



Managed Buffering of Jitter

FIG. 13

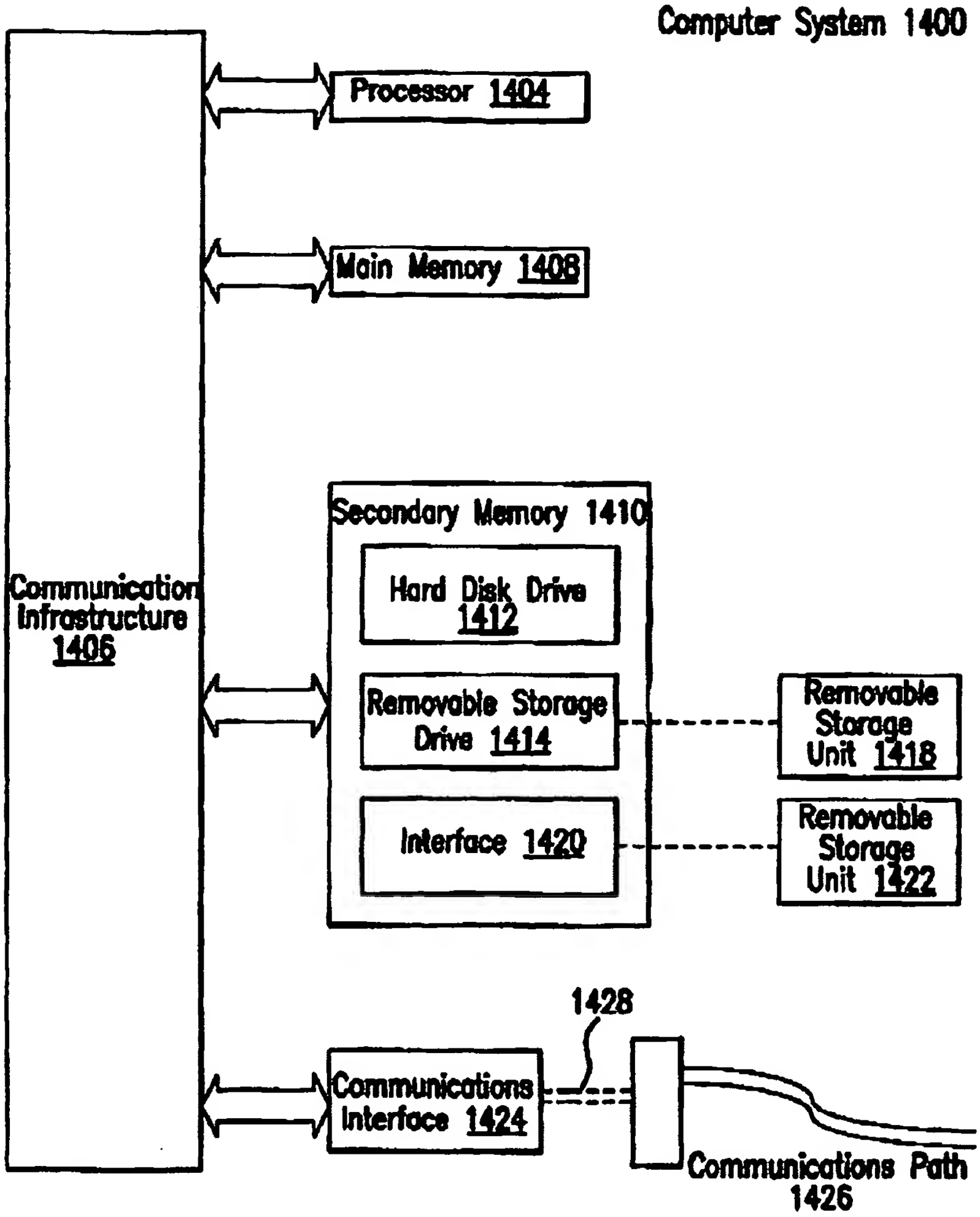

Computer System 1400
Processor 1404
Main Memory 1408
Communication Infrastructure 1406
Secondary Memory 1410
Hard Disk Drive 1412
Removable Storage Drive 1414
Interface 1420
Removable Storage Unit 1418
Removable Storage Unit 1422
1428
Communications Interface 1424
Communications Path 1426

FIG. 14

METHOD, SYSTEM, AND COMPUTER PROGRAM PRODUCT FOR MANAGING JITTER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention pertains to achieving optimal quality when transmitting voice data over a lossy network; more particularly, it pertains to managing jitter buffering of data packets over a packet-switched network.

2. Related Art

Latency and jitter are important aspects of network performance that can degrade communication between any two points on a packet-switched network, like the Internet. Latency is the delay introduced on packets during travel from one site to another. Latency will be perceived by the end users as a delay in the response of the remote site. Jitter is the variation in latency from one packet to another.

Latency and jitter each impact communication differently. F or example, if packets always arrived 50 milliseconds (ms) after being transmitted, then there would be a 50 ms latency and no jitter. In another example, however, if packet #1 arrived 100 ms after transmission, packet #2 arrived 50 ms after transmission, and packet #3 arrived 150 ms after transmission, there would be an average jitter of +/−33 ms. In voice over Internet protocol (VoIP) applications, jitter is more critical than latency. Jitter can cause a packet to arrive too late to be useful. The effect is that the packet may be delayed enough that the end user will hear a pause in the voice that is talking to them, which is very unnatural if it occurs during the middle of a word or sentence.

Jitter typically occurs when the network utilization is too high, and packets are being queued, causing delivery times to become unpredictable. The Internet, because of its complex structure, is often subject to varying degrees of jitter. Jitter variation can occur at different locations and at different times depending upon network traffic and other conditions. Thus, jitter needs to be managed.

Effective jitter management is especially needed in VoIP applications. Each VoIP call needs jitter management. FIG. **1** shows an example VoIP architecture **100**, including gateways **110**, **120** that provide an interface between public-switched telephone networks (PSTN) **130**, **140** and a packet-switched network **102**. A voice call is carried out between telephone **150** and telephone **160** through PSTN **130**, gateway **110**, network **102**, gateway **120**, and PSTN **140**.

Static jitter buffering is one conventional technique to compensate for jitter. As shown in FIG. **2**, static jitter buffering is carried out in gateway **120** which receives voice packets from network **102**. A static jitter buffer **220** is provided to buffer the received voice packets from network **102**. In such static jitter buffering, however, there is a compromise between the size of the jitter buffer and the delay of voice packets waiting in the jitter buffer. In particular, if the jitter buffer is large, it accommodates greater variation in jitter. The output packet traffic may not be jittery, but noticeable delays occur. If the jitter buffer is small, the delay is smaller but gaps in traffic are not accommodated.

SUMMARY OF THE INVENTION

A method, system, and computer program product is provided that manages jitter in packet-switched networks. In one embodiment, the present invention manages jitter in a VoIP system that includes a framer, a traffic analyzer, and a jitter manager. The framer time-stamps incoming packets and discards out-of-order packets. The framer outputs the in-order packets to the traffic analyzer and the jitter manager. The traffic analyzer maintains a sliding window array of a set of packets for use in calculating jitter statistics. These statistics are sent from the traffic analyzer to the jitter manager. The jitter manager uses these statistics to manage the flow of packets, the insertion or discardation of silence packets, and the supervision of any connected jitter buffers.

Handling jitter comes at the expense of latency, however, since the only way to handle jitter is to buffer additional data. So that when the data arrives exceptionally late, continuous playback to the end user can be maintained. Yet, the present invention manages the jitter buffer's size so that the latency does not grow too long. In this way, the present invention compensates for network jitter without resorting to excessive buffering.

Further embodiments, features, and advantages of the present invention, as well as the structure and operation of the various embodiments of the present invention, are described in detail below with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE FIGURES

The accompanying drawings, which are incorporated herein and form a part of the specification, illustrate the present invention and, together with the description, further serve to explain the principles of the invention and to enable a person skilled in the pertinent art to make and use the invention.

In the drawings:

FIG. **1** illustrates how a packet generally travels over a VoIP system.

FIG. **2** is a diagram of a static jitter buffering system.

FIG. **3** is a diagram of a jitter buffer managing system according to one embodiment of the present invention.

FIG. **4** is a diagram illustrating a jitter buffer managing system of FIG. **3** according to the present invention.

FIG. **5** is a diagram for a framing system of FIG. **4** according to the present invention.

FIG. **6** is a diagram of a method for framing in one example implementation of the present invention.

FIG. **7** is a diagram of a traffic analyzer of FIG. **4** according to the present invention.

FIG. **8** is a diagram of a method for analyzing traffic in one example implementation of the present invention.

FIG. **9** is a diagram of a jitter manager of FIG. **4** according to the present invention.

FIG. **10** is a diagram of a method for managing jitter in one example implementation of the present invention.

FIG. **11** is a diagram of the output of a gateway without jitter buffering.

FIG. **12** is a diagram of the output of a gateway with static jitter buffering.

FIG. **13** is a diagram of the output of the present invention with managed buffering.

FIG. **14** is an example computer system in one example implementation of the present invention.

The present invention will now be described with reference to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements. Additionally, the left-most digit(s) of a reference

number identifies the drawing in which the reference number first appears.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Table of Contents

I. Overview and Discussion

One shortfall of early VoIP systems was the poor quality of voice jittery voice) and the unacceptable latency caused by the fluctuating, and at times less than adequate bandwidth available through the Internet.

According to the present invention, jitter buffer managing is used to resolve the quality of voice over the unpredictable and some time limited bandwidth of the Internet. This capability adjusts the size and contents of the jitter buffer, thus minimizing jitter.

The present invention provides a method, system, and computer program product for managing jitter. In one embodiment, there are four basic components:

Framer

Traffic Analyzer

Jitter Manager

Jitter Buffer

These components are delineated only for explanation, and the features of each component can easily be incorporated into other components. In one example, these components can be implemented on a gateway server within a VoIP system described in co-pending U.S. patent application Ser. No. 09/393,658 (incorporated herein by reference in its entirety). However, the gateway server and reference are not intended to limit the present invention.

In one example, the present invention provides a method for achieving optimal quality when transmitting voice over a lossy network. The origin gateway indexes the outgoing packets. The framer time-stamps incoming packets and discards out-of-order packets. The traffic analyzer determines jitter statistics for the traffic of in-order packets output from the framer. The traffic analyzer communicates the jitter statistics to the jitter manager. The jitter manager coordinates the incoming traffic of in-order, time-stamped, indexed packets based on the jitter statistics from the traffic analyzer and the contents of the packets from the framer. The jitter manager inserts or discards silence packets, and maintains the jitter buffer.

In one implementation, a framer, traffic analyzer,jitter manager and jitter buffer can run on the same server or personal computer (PC). Alternatively, the functionality of the jitter buffer managing system can be carried out on physically separate machines. For example, a network could typically include a framer running on a gateway server. The traffic analyzer and jitter manager can be connected to the same network, but run on a different PC. The jitter buffer can be implemented in hardware or software.

II. Terminology

The term "traffic" refers to voice, facsimile, video, multimedia, digital information, or other data that can be sent between telephony terminal equipment and/or network terminal equipment.

The term "jitter statistics" refers to any of a number of statistics generated from the values calculated for jitter, jitter variation (also known as interpacket time or width, which reflect changes in the size of a packet from start to destination), average jitter, average jitter variation and any combination thereof.

The term "sliding window array" refers to a matrix or other data structure which can be filled with jitter statistics and updated.

III. Managed Jitter Buffering Embodiment

FIG. **3** shows an example VoIP architecture **300** that includes a gateway server **310** coupled to a PSTN **140**. According to the present invention, gateway server **310** includes a jitter buffer manager **320** coupled to jitter buffer **330**. Jitter buffer **330** represents any number of jitter buffers, static, dynamic or adaptive, implemented in hardware or software.

FIG. **4** is a diagram of a jitter buffer manager **320** according to an embodiment of the present invention. Jitter buffer manager **320**, among other things, minimizes the effects of packet loss, latency and packet degradation intrinsic to communication on packet-switched networks like the Internet.

Jitter buffer manager **320** includes a framer **410**, a traffic analyzer **420**, a jitter manager **430**, and a jitter buffer **330**. For example, framer **410** is coupled to traffic analyzer **420** and jitter manager **430**. Traffic analyzer **420** is coupled to jitter manager **430**. Jitter manager **430** is coupled to jitter buffer **330**. Each of these components can run on the same PC or on separate PCs over a network.

An overview of each of the components of jitter buffer manager **320** is now provided. One example of a framer is shown in FIG. **5**. Framer **410** includes an input port **510**, a session clock **520**, a system clock **530**, a packet switch **540**, a discard buffer **550**, and an output port **560**. For example, an input port **510** is coupled to a session clock **520**. A system clock **530** is coupled to a session clock **520**. A session clock **520** is coupled to a packet switch **540**. A packet switch is coupled to a discard buffer **550** and an output port **560**.

For clarity, the operation of framer **410** is further described with respect to routine for framing **600** (FIG. *6*). Input port **510** receives network traffic as indexed packets (step **620**). Packets can be indexed in numerous ways. It is well-known in the field of the present invention that headers can be added to packetized data. These headers can contain routing information, time-stamps, and other information. Here, an index is added by the origin gateway **110**. Session clock **520** time-stamps each indexed packet upon its arrival (step **630**) to produce time-stamped, indexed packets. Session clock **520** maintains its clock through a connection to the system clock **530**. The system clock **530** can be hardware or software, resident or maintained on another PC.

Packet switch **540** carries out steps **650–660**. In step **650**, the time-stamp and index of each packet is checked to determine whether the packet arrived out-of-order. If the packet arrived out-of-order, it is discarded (step **660**). Otherwise, in step **670**, output port **560** sends the remaining packets to the traffic analyzer **420** and jitter manager **430**. At this point, the traffic is a stream of in-order, time-stamped, indexed packets. Routine **600** was described above with respect to the example framer **410** shown in FIG. **5**. This is not intended to limit the present invention. Other embodiments can be used as would be apparent to a person skilled in the art given this description.

One example of a traffic analyzer is shown in FIG. **7**. The traffic analyzer **420** includes an input port **710**, a calculator **720**, a sliding window **730**, and an output port **740**. For

example, input port **710** is coupled to calculator **720**. Calculator **720** is coupled to sliding window array **730** and output port **740**. For clarity, the operation of the traffic analyzer **420** is further described with respect to routine for analyzing traffic **800** (FIG. **8**). Input port **710** receives traffic (step **810**) from the framer **410**. In one embodiment, the traffic is a stream of in-order, time-stamped, indexed packets. Calculator **720** calculates the jitter (step **820**) and jitter variation (step **830**) for each received packet (step **810**). For example, one way of calculating jitter is to take the absolute value of the difference between the actual interpacket time and the theoretical interpacket time. The interpacket time is width in terms of time of a packet. For instance, a 30 ms packet has a theoretical interpacket time of 30 ms. This same packet may not arrive at the destination gateway **120** with the same interpacket time. Thus, jitter is the difference between the actual or received interpacket time and its theoretical value.

Similarly,jitter variation can be calculated (step **830**). In one embodiment of the present invention, if the sliding window array **730** is empty, then jitter variation is considered to be zero. Otherwise, average jitter is calculated using the sliding window array **730**, and jitter variation is the absolute value of the difference between the present jitter and average jitter. In one example, the average jitter is calculated by summing the jitter values over a number of jitter points. More specifically, the sliding window array **730** stores the jitter, jitter variation for the last Ns packets (Ns is a variable). J[1] refers to most recently stored jitter value, J[Ns] refers to the oldest jitter value that is still stored. Similarly, JV[1] refers to most recently stored jitter variation value, JV[Ns] refers to the oldest jitter variation value that is still stored. Updating the sliding window (step **850**) consists of shifting J[1] into J[2], J[2] into J[3], and storing the new value in J[1]. The value previously stored in J[Ns] will be lost in this process. The same procedure is used to update JV values. The sliding window array **730** stores the jitter and jitter variation (step **840**). The sliding window array **730** is updated with these jitter statistics for each packet (step **850**).

In one embodiment, the average jitter is calculated (step **860**) by computing Jave=(Cw[1]xJ[1]+Cw[2]xJ[2]+ . . . +Cw[Ns]xJ[Ns])/(Cw[1]+Cw[2]+ . . . +Cw[Ns]). For this embodiment, Cw[1 . . . Ns] are co-efficients that are used to give more weighting to certain packets in relation to one another within the sliding window array **730**. The same procedure is used to compute JVave (step **870**). Thus, calculator **720** calculates an average value for jitter and jitter variation (steps **860–870**) and updates the sliding window with these values (step **880**). In one embodiment, these values are outputted (step **890**) via output port **740** to the jitter manager **430**. Routine **800** was described above with respect to the example traffic analyzer shown in FIG. **7**. This is not intended to limit the present invention. Other embodiments can be used as would be apparent to a person skilled in the art given this description.

One example of a jitter manager is shown in FIG. **9**. The jitter manager **430** includes an input port **910**, an update port **920**, a calculator **930**, a packet switch **940**, a silence packet generator **950**, and an output port **960**. For example, an input port **910** is coupled to a calculator **930**. An update port **920** is coupled to calculator **930**. Calculator **930** is coupled to packet switch **940**. Silence packet generator **950** is coupled to packet switch **940**. Packet switch **940** is coupled to output port **960**. For clarity, the operation of the jitter manager **430** is further described with respect to routine for managing jitter **1000** (FIG. **10**). Input port **910** receives traffic from the framer **410** (step **1005**). In one embodiment, the traffic is in-order, time-stamped, indexed packets. Input port **910** sends the traffic to calculator **930**. Calculator **930** also receives the jitter statistics from the update port **920** (step **1015**). Calculator **930** calculates the target jitter buffer size (step **1010**). In step **1020**, each packet is checked to see if it contains silence data. If the packet does contain silence data, then, in one embodiment, the calculator **930** checks the current jitter buffer size (step **1025**). The calculator **930** re-checks the target jitter buffer size (step **1010**) using the jitter statistics (step **1015**). In one example, the target jitter buffer size (step **1010**) can then be calculated as: Jt=Jc+(Cjx Jave)+(CvxJVave). Jt is the target jitter buffer size. Jc is a jitter constant, representing the minimum possible target buffer size. Cj is the jitter co-efficient, adjusting how much observed jitter will be reflected in the target jitter buffer. Cv is the jitter variation co-efficient, adjusting how much observed jitter variation will be reflected in the target jitter buffer. In one embodiment, these values can be predetermined: Cw[1]=Cw[n]=Cw[Ns]=1, Jc=30 ms, Cj=1, and Cv=2.

In step **1040**, the packet switch **940** compares the current actual jitter buffer size with the determined target jitter buffer size. If the actual jitter buffer size is larger than the target jitter buffer size, then the silence packet is discarded (step **1045**). If the actual jitter buffer size is equal to or smaller than the target jitter buffer size, then the silence packet is inserted into the jitter buffer **330** (step **1050**). If the packet does not contain silence data (step **1040**), then, in one embodiment, the packet switch **940** checks to see if the jitter buffer **330** is empty (step **1030**). If the jitter buffer **330** is empty, then the packet is inserted into the jitter buffer **330** (step **1035**). If the jitter buffer **330** is not empty, then the calculator **930** checks the jitter buffer **330** as discussed above (step **1025**).

The calculator **930** also compares the actual jitter buffer size with the target jitter buffer size (step **1055**). In one embodiment, if the actual jitter buffer size is smaller than the target jitter buffer size, then a silence packet is inserted into jitter buffer **330** (step **1060**). The packet switch **940** obtains silence packets from silence packet generator **950**. The generation of silence packets is well-known in the field of the present invention. There was many ways to create packets and insert them into network traffic. Generating packets without any voice data is similarly straightforward. If the actual jitter buffer size is equal to or larger than the target jitter buffer size, then the packet switch **940** inserts the packet into jitter buffer **440** (step **1035**).

Routine **1000** was described with respect to the example jitter manager in FIG. **9**. This is not intended to limit the present invention. Other embodiments can be used as would be apparent to a person skilled in the art given this description.

FIGS. **11–13** show various outputs of VoIP system. FIG. **11** shows the unbuffered output of gateway **120** to PSTN **140**. FIG. **12** shows the output from a static jitter buffer **220** to PSTN **140** in FIG. **2**. FIG. **13** shows the output from the jitter buffer **330** of FIG. **3**, where an embodiment of the present invention is shown as the jitter buffer manager **320**. Each of the figures shows the arrival of the same set of voice and silence packets along the left-hand column. Time is displayed in 30 ms segments. The packets have an interpacket time of 30 ms. T he packets are numbered for illustrative purposes here, and the actual indexing of traffic may take other forms. For example, the index may start at zero (0) and count up until a silence packet is encountered by gateway **110**. At this time, gateway **110** gives the next

voice packet an index of zero (0) and repeats the process. Gaps are denoted in FIGS. 11–13 by a "*" symbol. The gaps and bursts in arrival time of the uniformly sent packets is illustrative of network congestion in IP 102.

In these examples, packets #1 and #2 arrive on time. Packets #1 and #2 are followed by a gap, and then the delayed packets #3 and #4. Packets #5 and #6 arrive on time. Packets #5 and #6 are followed by another gap. Packets #7 and #8 arrive at almost the same time as packets #9 and #10. These packets are immediately followed by packets #11 and #12. This pattern of received packets is exemplary of the results of network congestion. These different outputs show in the right-hand column of these figures. The outputs are discussed in detail below.

FIG. 11 shows the loss of packets #5, #9 and #10. Network congestion and the lack of any buffering to retain these packets caused the system to lose them. The resulting output has gaps or breaks in conversation. This is not a desirable result as the flow of the conversation is compromised. Traditionally, this problem was alleviated by providing a static buffer.

FIG. 12 shows the output from static buffering. Here, the static buffer receives the same data as in FIG. 11, but waits 30 ms before playing the packets out. The packets are held in the buffer. This means that communications are delayed 30 ms (or one packet) each time the buffer is empty. Typically, a buffer can hold 300 ms (or 10 packets), but other configurations are possible. The buffer plays out the packets until it is empty. The gap after packet #2 is small enough that the buffer can cover it. The gap after packet #6 is too large for the buffer to cover. The result is a gap or break in the packet flow, which is interpreted as a break in the conversation. The static buffer holds packet #7 for 30 ms to regain some buffering. Although not shown here, a buffer can be configured to play out packets without any delay in the event of network congestion similar to that experienced in packets #6–#12. In such a configuration, the buffer would only be used to prevent packet loss during a burst.

FIG. 13 shows the output from a managed buffer taught by the present invention. The jitter buffer manager system 320 receives the same data as in FIG. 11 and FIG. 12. The jitter buffer manager system 320 is discussed in detail above. Among other things, the jitter buffer manager system 320 includes a packet switch 940. Packet switch 940 performs, among other things, steps 1020, 1030, 1040, and 1055. These steps perform the insertion and deletion of packets when the jitter buffer is smaller or larger than a target jitter buffer size. The target jitter buffer size is calculated based on the jitter statistics. In FIG. 13, the output of the managed buffer shows the insertion of a silence packet (Is) after packet #4. The silence packet is inserted when the actual jitter buffer size is smaller than the target jitter buffer size. This situation exists after the arrival of packets #3 and #4. Packets #3 and #4 arrive after a gap. The gap reduces the actual jitter buffer size below the target jitter buffer size. Thus, a silence packet is generated and inserted (step 1060). The insertion of the silence packet closes the gap in the packets which was present in both FIG. 11 and FIG. 12. With respect to the second burst, the jitter buffer manager 320 can be configured to play out packets immediately if jitter buffer 330 is empty (steps 1030 and 1035). Subsequently, the jitter buffer manager 320 would insert data and silence packets according to the steps of FIG. 10 to maintain the target jitter buffer size.

The advantages of the present invention are provided by the ability of the jitter buffer manager 320 to maintain jitter buffer 330 in such a way that the outputted traffic is continuous. Moreover, when the traffic is a stream of packets, the present invention maintains the coherency and quality of the voice data being outputted.

Example Computer System

An example of a computer system 1400 is shown in FIG. 14. Computer system(s) 1400 can execute software to carry out any of the functionality described above with respect to jitter buffer manager system 320, including any of the components 410–430.

Computer system 1400 represents any single or multiprocessor computer. Single-threaded and multi-threaded computers can be used. Unified or distributed memory systems can be used.

Computer system 1400 includes one or more processors, such as processor 1404. One or more processors 1404 can execute software implementing all or part of jitter manager system 400 as described above. Each processor 1404 is connected to a communication infrastructure 1402 (e.g., a communications bus, cross-bar, or network). Various software embodiments are described in terms of this exemplary computer system. After reading this description, it will become apparent to a person skilled in the relevant art how to implement the invention using other computer systems and/or computer architectures.

Computer system 1400 also includes a main memory 1408, preferably random access memory (RAM), and can also include secondary memory 1410. Secondary memory 1410 can include, for example, a hard disk drive 1412 and/or a removable storage drive 1414, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive 1414 reads from and/or writes to a removable storage unit 1418 in a well known manner. Removable storage unit 1418 represents a floppy disk, magnetic tape, optical disk, etc., which is read by and written to by removable storage drive 1414. As will be appreciated, the removable storage unit 1418 includes a computer usable storage medium having stored therein computer software and/or data.

In alternative embodiments, secondary memory 1410 may include other similar means for allowing computer programs or other instructions to be loaded into computer system 1400. Such means can include, for example, a removable storage unit 1422 and an interface 1420. Examples can include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units 1422 and interfaces 1420 which allow software and data to be transferred from the removable storage unit 1422 to computer system 1400.

Computer system 1400 can also include a communications interface 1424. Communications interface 1424 allows software and data to be transferred between computer system 1400 and external devices via communications path 1426. Examples of communications interface 1424 can include a modem, a network interface (such as Ethernet card), a communications port, etc. Software and data transferred via communications interface 1424 are in the form of signals which can be electronic, electromagnetic, optical or other signals capable of being received by communications interface 1424, via communications path 1426. Note that communications interface 1424 provides a means by which computer system 1400 can interface to a network such as the Internet.

The present invention can be implemented using software running (that is, executing) in an environment similar to that

described above with respect to FIG. **14**. In this document, the term "computer program product" is used to generally refer to removable storage unit **1418**, a hard disk installed in hard disk drive **1412**, or a carrier wave or other signal carrying software over a communication path **1426** (wireless link or cable) to communication interface **1424**. A computer useable medium can include magnetic media, optical media, or other recordable media, or media that transmits a carrier wave. These computer program products are means for providing software to computer system **1400**.

Computer programs (also called computer control logic) are stored in main memory **1408** and/or secondary memory **1410**. Computer programs can also be received via communications interface **1424**. Such computer programs, when executed, enable the computer system **1400** to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable the processor **1404** to perform the features of the present invention. Accordingly, such computer programs represent controllers of the computer system **1400**.

In an embodiment where the invention is implemented using software, the software may be stored in a computer program product and loaded into computer system **1400** using removable storage drive **1414**, hard drive **1412**, or communications interface **1424**. Alternatively, the computer program product may be downloaded to computer system **1400** over communications path **1426**. The control logic (software), when executed by the one or more processors **1404**, causes the processor(s) **1404** to perform the functions of the invention as described herein.

In another embodiment, the invention is implemented primarily in firmware and/or hardware using, for example, hardware components such as application specific integrated circuits (ASICs). Implementation of a hardware state machine so as to perform the functions described herein will be apparent to persons skilled in the relevant art(s).

IV. Conclusion

While specific embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. It will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined in the appended claims. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A system that manages jitter over a packet-switched network, comprising:

a framer for appending a timestamp to a received indexed packet;

a traffic analyzer coupled to said framer, wherein said traffic analyzer examines the received indexed packet to determine whether the received indexed packet arrived in a proper sequential order and computes jitter statistics;

a jitter manager coupled to said framer and to said traffic analyzer, wherein said jitter manager uses the jitter statistics to control the flow of the received indexed packet; and

a jitter buffer coupled to said jitter manager for receiving the received indexed packet or a silence packet from said jitter manager, wherein a size of said jitter buffer is controlled by said jitter manager.

2. A system of claim **1**, further comprising a sliding window array, wherein said sliding window array stores jitter measurements for received indexed packets.

3. A system of claim **1**, wherein the jitter statistics computed by said traffic analyzer include average jitter variation for at least three received data packets.

4. A system of claim **1**, wherein the jitter statistics computed by said traffic analyzer include average jitter for at least two received data packets.

5. A system that manages jitter, comprising:

an input port that receives traffic;

an update port capable of receiving jitter statistics for the traffic,

wherein the jitter statistics comprise average jitter and average jitter variation for the traffic;

a calculator coupled to said input port and said update port that receives jitter statistics from said update port and that calculates the target size for a jitter buffers; and

a packet-switch that inserts traffic into the jitter buffers.

6. A method of connecting, comprising the steps of:

handling the traffic from a packet-switched network to a public switched telephone network; and

managing the jitter of said traffic, wherein a manager adjusts an adaptive buffer to regulate the flow of data packets, wherein a size of the adaptive buffer is determined based on jitter statistics comprising a jitter measurement for a message currently being processed, an average jitter of a set of received data packets and the average jitter variation of the set of received data packets.

7. A method for managing the flow of a data packet within a set of data packets, comprising the steps of:

(a) receiving the data packet, wherein the received data packet is the most recent data packet received within the set of data packets;

(b) measuring a jitter for the data packet received in step (a);

(c) computing a jitter variation for the data packet received in step (a);

(d) computing an average jitter measurement for the set of data packets;

(e) computing an average jitter variation for the set of data packets; and

(f) computing a target buffer size, wherein the target buffer size is a function of the average jitter measurement and the average jitter variation.

8. The method of claim **7**, wherein said step (c) comprises taking the absolute difference between the jitter measurement calculated in step (b) and an average jitter measurement for the set of data packets that does not include the jitter measurement calculated in step (b).

9. The method of claim **7**, wherein computing said average jitter in step (d) comprises computing the summation of C_i times J_i over N data packets within the set of received data packets, wherein i varies from 1 to N, C_i equals a jitter weighting coefficient, J_i equals a jitter measurement for the i^{th} received data packet, and N is a positive integer.

10. The method of claim **7**, wherein said step (d) includes applying greater weighting to measurements associated with more recently received data packets with a set of Ci jitter weighting coefficients.

11. The method of claim **7**, further comprising:

(g) timestamping the received data packet; and

(h) when the received data packet is not in proper sequential order, discarding the received data packet.

12. The method of claim 7, further comprising:

(g) when a buffer size is less than the target buffer size, inserting silence packets into the buffer.

13. The method of claim 7, further comprising:

(g) when a buffer size is greater than the target buffer size, removing silence from a received data packet.

14. A method for calculating the size of a buffer used to manage the flow of a set of received data packets, comprising the steps of:

(a) establishing a minimum buffer size;

(b) computing an average jitter measurement based on at least two received data packets;

(c) computing an average jitter variation measurement based on at least two received data packets; and

(d) determining a target buffer size based on the values determined in step (a), step (b) and step (c).

15. The method of claim 14, wherein step (d) comprises adding the minimum buffer size established in step (a) to the product of an average jitter coefficient times the average jitter measurement determined in step (b) and adding the product of an average jitter variation coefficient times the average jitter variation measurement determined in step (c).

16. The method of claim 14, wherein computing the average jitter measurement in step (b) comprises computing the summation of C_i times J_i over N received data packets within the set of received data packets, wherein i varies from 1 to N, C_i equals a jitter weighting coefficient, J_i equals a jitter measurement for the i^{th} received data packet, and N is a positive integer.

17. The method of claim 16, wherein said step (b) includes applying greater weighting to measurements associated with more recently received data packets with a set of Ci jitter weighting coefficients.

18. The method of claim 14, further comprising:

(e) maintaining a sliding window array that stores jitter measurements for received data packets, wherein said maintaining step includes adjusting the size of the sliding window array.

19. The method of claim 14, further comprising:

(e) maintaining a sliding window array that stores jitter measurements for received data packets, wherein said maintaining step includes adjusting the size of the sliding window array.

* * * * *

US006377931B1

(12) **United States Patent**
Shlomot

(10) **Patent No.: US 6,377,931 B1**
(45) **Date of Patent: Apr. 23, 2002**

(54) **SPEECH MANIPULATION FOR CONTINUOUS SPEECH PLAYBACK OVER A PACKET NETWORK**

(75) Inventor: **Eyal Shlomot**, Irvine, CA (US)

(73) Assignee: **Mindspeed Technologies**, Newport Beach, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/407,466**

(22) Filed: **Sep. 28, 1999**

(51) **Int. Cl.**[7] **G10L 21/04**; G01R 29/26; G11B 7/007

(52) **U.S. Cl.** **704/503**; 704/278; 369/44.32; 702/69

(58) **Field of Search** 704/201, 500, 704/503, 278; 370/506, 522; 709/219; 369/44.32; 702/69

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,694,521	A	12/1997	Shlomot et al.	395/2.71
5,699,481	A	12/1997	Shlomot et al.	395/2.37
5,825,771	A *	10/1998	Cohen et al.	370/522
5,881,245	A *	3/1999	Thompson	709/219
5,953,695	A *	9/1999	Barazesh et al.	704/201
6,212,206	B1 *	4/2001	Ketcham	370/506

OTHER PUBLICATIONS

Overview of Speech Packetization, M.H. Sherif and A. Crossman, AT&T Bell Laboratories, © 1995 IEEE, pp. 296–304.

Ansari et al ("Compressed Voice Integrated Services Frame Relay Networks: Voice Synchronization," Conference on Electrical and Computer Engineering, p. 1073–1076 vol. 2, Sep. 5–8, 1995).*

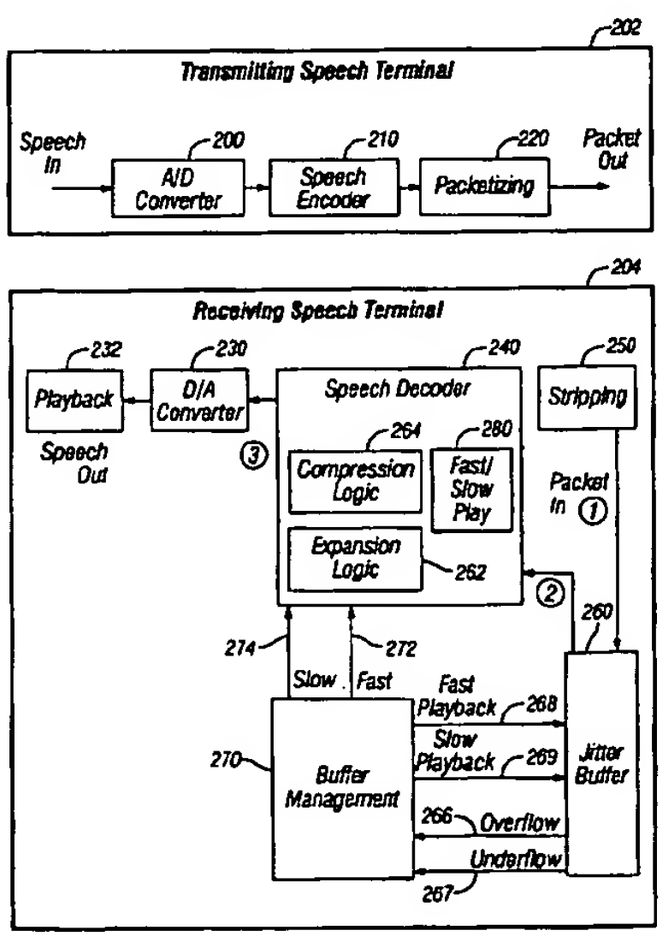
* cited by examiner

Primary Examiner—Richemond Dorvil
Assistant Examiner—Daniel A. Nolan
(74) *Attorney, Agent, or Firm*—Akin, Gump, Strauss, Hauer & feld, LLP

(57) **ABSTRACT**

In a speech communications network, continuous play of audio packets is achieved using a jitter buffer in a receiver. Audio packets are stored in the jitter buffer before decoding the audio packets into an audible output. When the level of stored audio packets approaches the full capacity of the jitter buffer, the rate at which the audio packets are played out of the jitter buffer is increased signaling a compression operation in the decoder. When the level of stored audio packets approaches an empty level of the jitter buffer, the rate which the audio packets are played out of the jitter buffer is reduced signaling an expansion operation in the decoder. Audio packets are not modified when the level of stored audio packets is within a predetermined range. A speed controller is provided to instruct the decoder to decode the audio packets according to either a compressed, expanded or normal audio packet status.

20 Claims, 4 Drawing Sheets



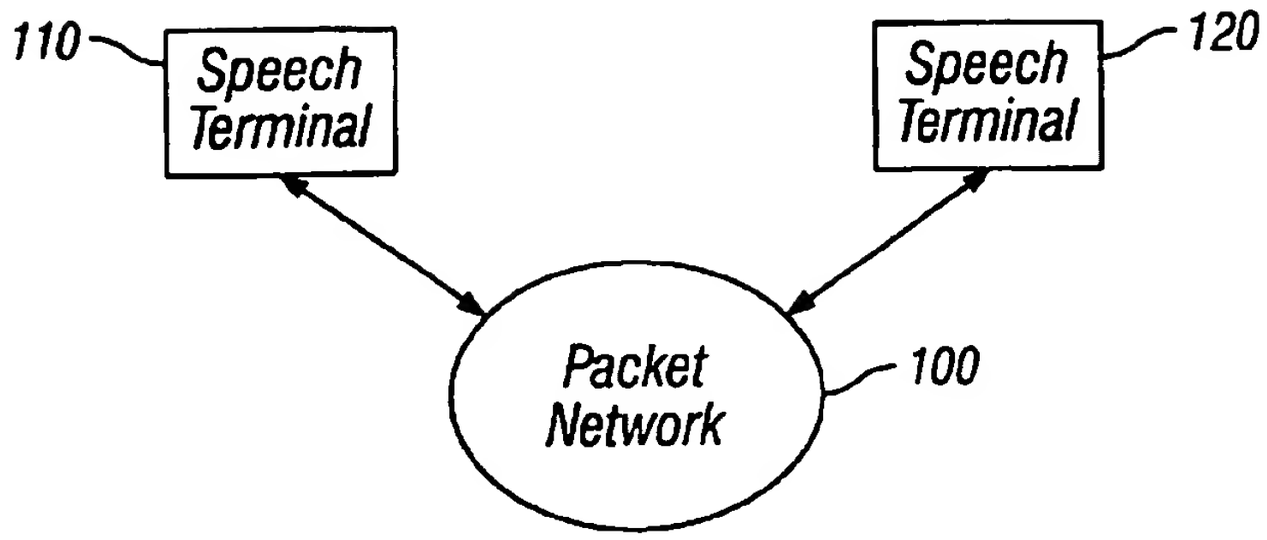


FIG. 1
(Prior Art)

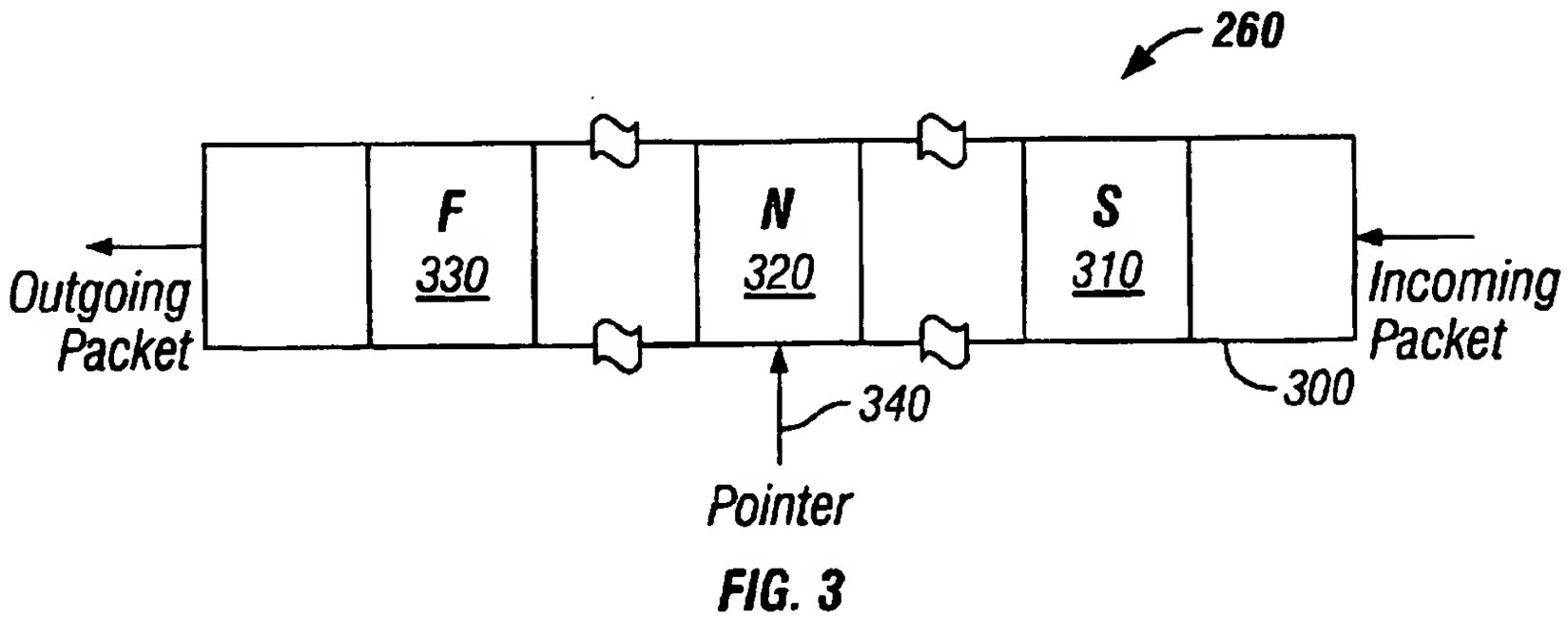


FIG. 3

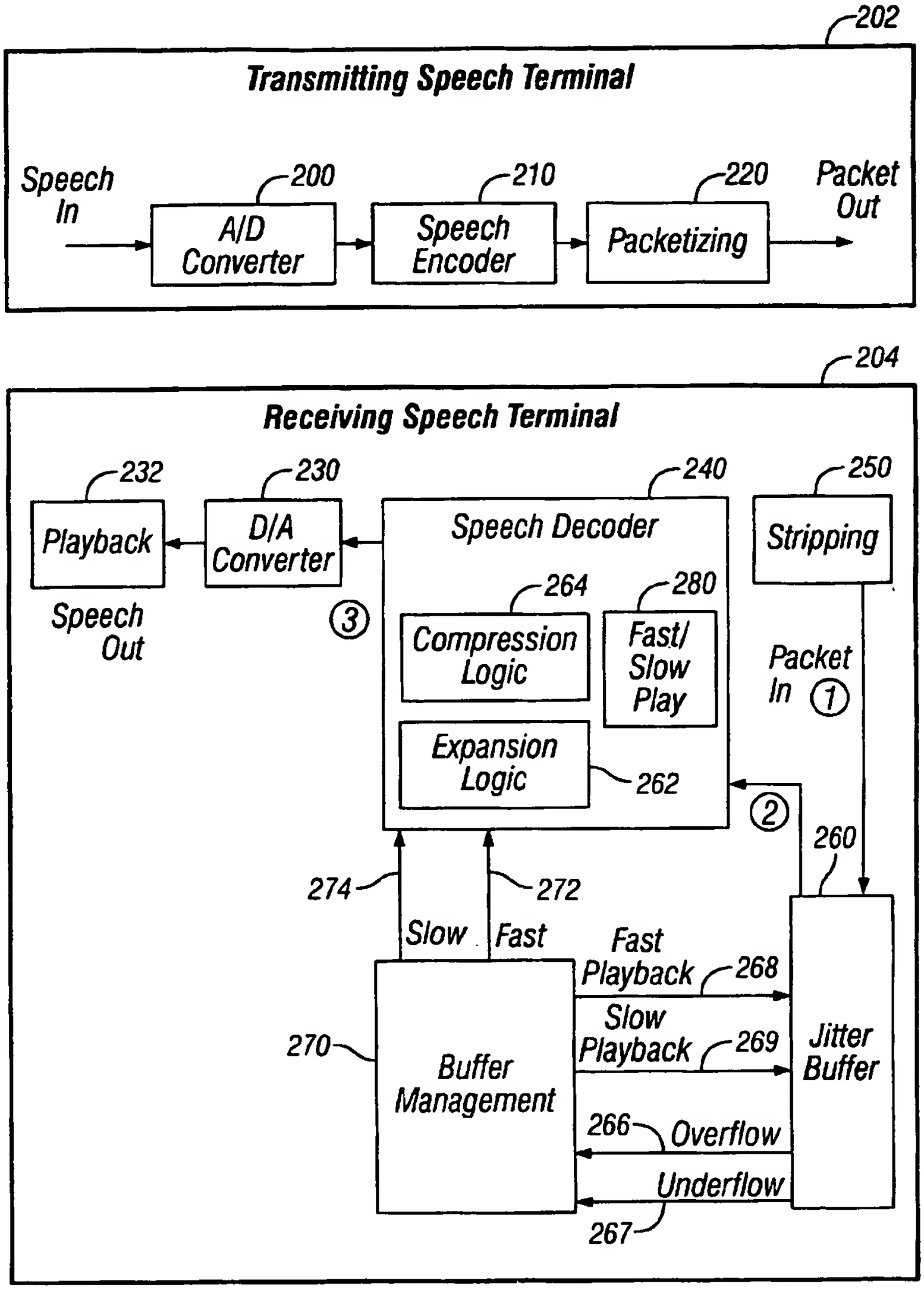

202
Transmitting Speech Terminal
Speech In
200
A/D Converter
210
Speech Encoder
220
Packetizing
Packet Out
204
Receiving Speech Terminal
232
Playback
230
D/A Converter
240
Speech Decoder
250
Stripping
Speech Out
3
264
Compression Logic
280
Fast/ Slow Play
Expansion Logic
262
Packet In 1
2
260
274
272
Slow
Fast
Fast Playback
268
Slow Playback
269
270
Buffer Management
Jitter Buffer
266
Overflow
Underflow
267

FIG.2

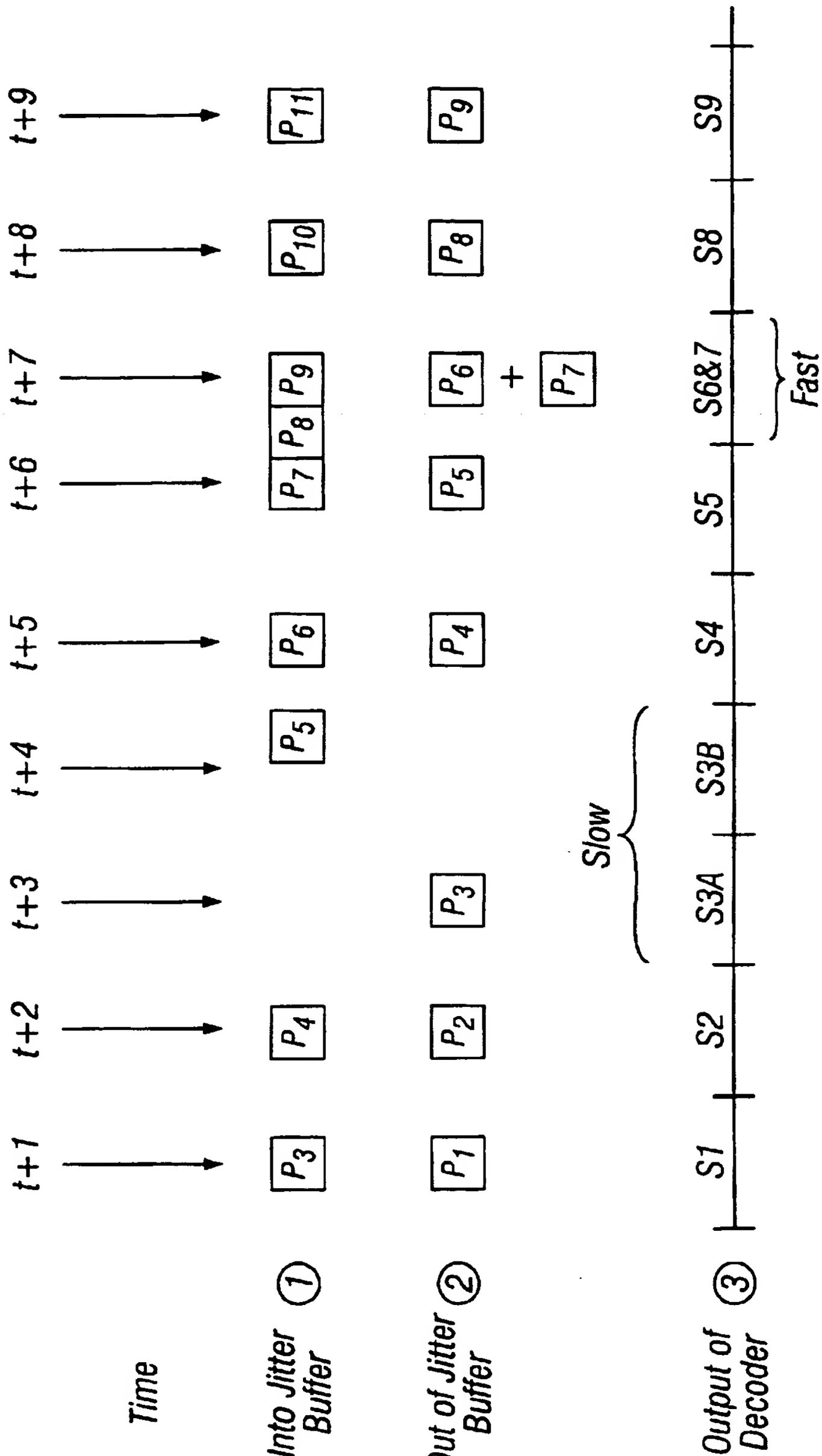

t+9
t+8
t+7
t+6
t+5
t+4
t+3
t+2
t+1
P11
P10
P9
P8
P7
P6
P5
P4
P3
P2
P1
P9
P8
P6
+
P7
P5
P4
P3
Time
Into Jitter Buffer ①
Out of Jitter Buffer ②
Output of Decoder ③
S9
S8
S6&7
Fast
S5
S4
S3B
S3A
Slow
S2
S1

FIG. 4A

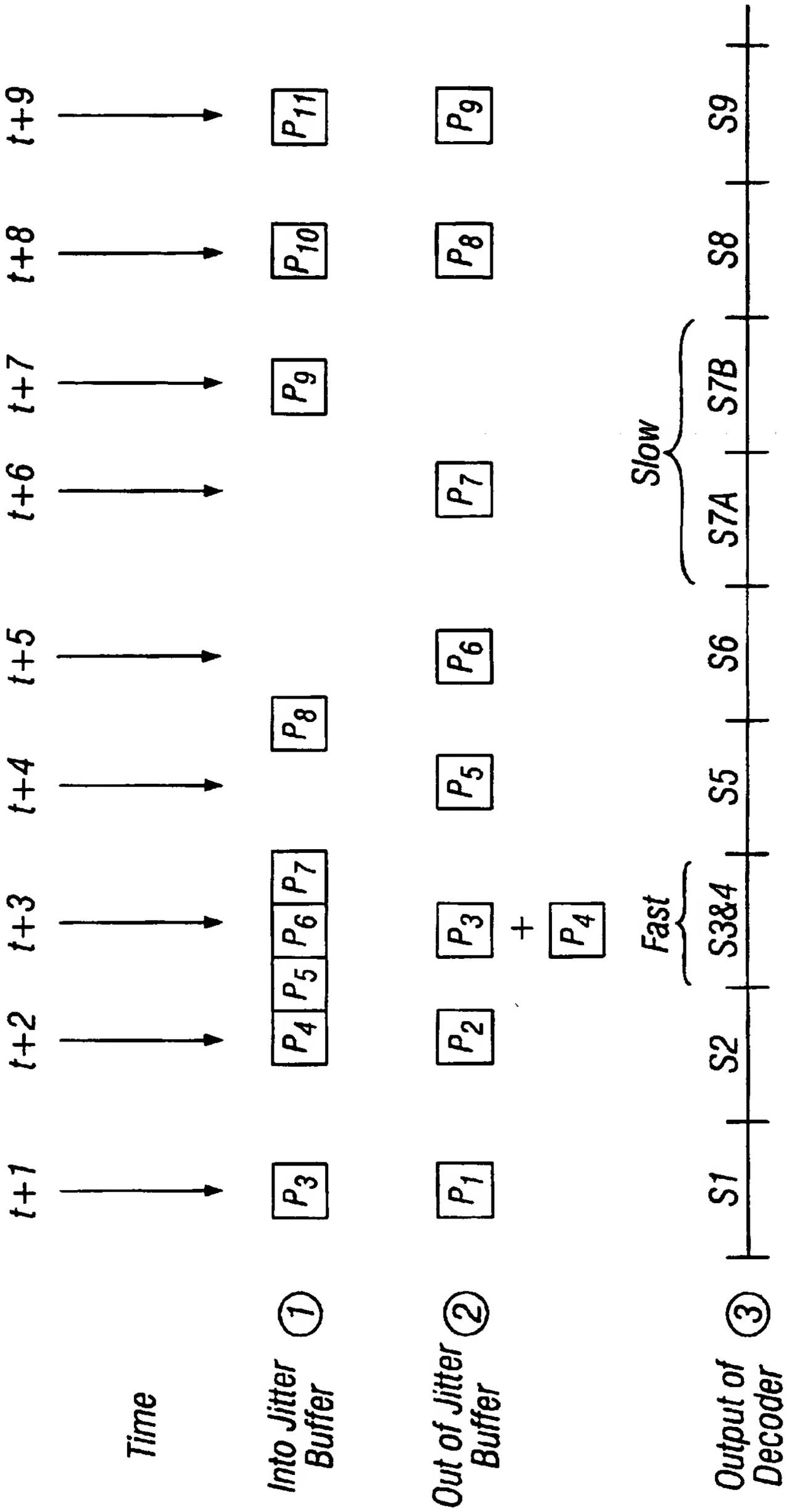

Time
Into Jitter Buffer ①
Out of Jitter Buffer ②
Output of Decoder ③
t+1
t+2
t+3
t+4
t+5
t+6
t+7
t+8
t+9
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
+
Fast
Slow
S1
S2
S3&4
S5
S6
S7A
S7B
S8
S9

FIG. 4B

SPEECH MANIPULATION FOR CONTINUOUS SPEECH PLAYBACK OVER A PACKET NETWORK

BACKGROUND

1. Field of the Invention

The present invention relates to communication systems and in particular to packet network communication systems.

2. Description of the Related Art

Currently, global and local communication systems are rapidly changing from switched network systems to packet network systems. Packet network systems transmit data, speech, and video. An example of a packet network is the Internet (a globally connected packet network system) or the Intranet (a local area packet network system). While speech communications in switched network systems is carried by a direct point-to-point connection, speech communications in packet network system is performed by packing speech frames and transmitting the frames over the network.

A number of applications for packet networks now exist. For example, in November 1996, the International Telecommunication Union (ITU) and the Telecommunication Standardization Sector (ITU-T) ratified the H.323 specification defines how delay-sensitive voice and video traffic is transported over local area networks. Earlier this year (1999), the ITU-T approved H.323 Revision 2 for use in wide area networks. However, operating H.323 terminals over a wide area network (such as the public Internet) may result in poor performance due to the lack of quality-of-service (QoS) guarantees in packet networks. In the Internet, congestion due to inadequate bandwidth often leads to long delays in the delivery of time-sensitive packets. For voice data, packets that are lost or discarded result in gaps, silence, and clipping in real-time audio playback.

To support a real-time QoS, a new Internet Protocol (IP) network has been proposed, called the Resource Reservation Protocol (RSVP). Using RSVP, both real time and non-real time applications can specify an appropriate QoS over the shared bandwidth of the Internet. However, until an RSVP standard is ratified and implemented in network routers, it is not possible for the end-to-end connections over IP networks to guarantee a QoS equivalent to the PSTN. In addition, IP telephony devices utilize Voice Over Internet Protocol (VOIP) over private and public carrier IP networks (rather than the public Internet) where ample bandwidth can be allocated.

Several drawbacks can jeopardize the quality of the speech transmitted by a packet network. The main drawback is the irregularity (or jitter) in the time of arrival of the packets. Since speech communications is a continuous process, each packet should be available at the receiving end in time for its usage (a packet is used by decoding its content and playing the decoded speech to the listener). A problem arises, for example, if a few packets are delayed at a node of the packet network. At the receiving end, since the speech packets have not arrived, the listener will experience a discontinuity in speech. Moreover, when the packets finally arrive to their destination, they might arrive too late to be used, and will be dropped. In this case, the listener will lose some of the speech information.

One possible solution for the irregular time of arrival of speech packets has been the buffering of several speech packets before using them to produce the speech. The speech packets are put in a FIFO (First-In-First-Out) buffer type, which holds several packets. Such a buffer is commonly

called a jitter buffer. If the number of delayed packets is less than the size of the buffer, then the buffer will not become empty, and the listener will not experience speech discontinuity or lost. The greater the potential jitter, the larger the buffer has to be, in order to give more room for the playback of previous packets while waiting for the subsequent arrival of later packets. However, the intermediate buffer does introduce an overall delay that is proportional to the buffer size.

A large size jitter buffer can overcome several irregularities in packet arrival time, but results in intolerable delay, while a small size jitter buffer introduces only a small delay, but recovers only a limited level of packet time-of-arrival jitter. The proper jitter buffer size is a system design concern, which should be determined according to the allowable speech communications delay, the expected network delays, and the tolerable reduction in speech quality due to discontinuities and losses.

Packet loss leads to unpleasant signal degradation. Small amounts of packet loss have been dealt with in a number of manners. One solution has been to employ packet replay, where the receiver merely repeats the last packet to fill in the time until the next packet actually arrives. However, where packet loss may be more substantial, such as where a Voice Over Internet Protocol (VOIP) signal passes over the Internet, simple packet replay has not been effective.

Another solution to minimize delay caused by a jitter buffer has been to dynamically monitor the jitter and adjust the buffer size accordingly. Commonly assigned U.S. Pat. No. 5,699,481 proposes the management of a jitter buffer by tracking the current number of speech packets stored in the jitter buffer. When the buffer approaches its full capacity, packets are removed from the jitter buffer. When the buffer approaches its empty level, "artificial" packets are inserted into the jitter buffer.

SUMMARY OF THE INVENTION

In a speech communications network, continuous play of received audio packets is achieved using a jitter buffer in a receiver. Audio packets are first temporarily stored in the jitter buffer before decoding of the audio packets into an audible output. A consistent accumulation level of the received audio packets in the jitter buffer is maintained to provide continuous and synchronized output to a decoder. When the level of stored audio packets approaches the full capacity of the jitter buffer, the rate at which the audio packets are played out of the jitter buffer is increased. The increased output rate is achieved by compressing a portion of the stored audio packets to reduce the number of audio packets in the jitter buffer. When the level of stored audio packets approaches an empty level of the jitter buffer, the rate which the audio packets are played out of the jitter buffer is reduced. The reduced output rate is achieved by expanding a portion of the stored audio packets to increase the number of audio packets in the jitter buffer. Audio packets are not modified when the level of stored audio packets is within a predetermined range, such that the rate of incoming audio packets received by the jitter buffer approximately equals the rate of decoded audio packets. A speed controller is then provided to instruct the decoder to decode the audio packets from the jitter buffer according to either a compressed, expanded or normal audio packet status.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained when the following detailed description of the

preferred embodiment is considered in conjunction with the following drawings, in which:

FIG. 1 is a block diagram of an exemplary speech communication packet network;

FIG. 2 is a block diagram of a transmitting speech terminal and a receiving speech terminal;

FIG. 3 is a block diagram of an exemplary jitter buffer structure of FIG. 2; and

FIGS. 4*a* and 4*b* are timing illustrations for packets communicated over the speech communication packet network of FIG. 1 and FIG. 2.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

The following related patent applications are hereby incorporated by reference as if set forth in their entirety:

U.S. Pat. No. 5,699,481, entitled "TIMING RECOVERY SCHEME FOR PACKET SPEECH IN MULTIPLEXING ENVIRONMENT OF VOICE WITH DATA APPLICATIONS," granted on Dec. 16, 1997 to Eyal Shlomot, et. al.; and

U.S. Pat. No. 5,694,521, entitled "VARIABLE SPEED PLAYBACK SYSTEM," granted on Dec. 2, 1997 to Eyal Shlomot, et. al.

The illustrative system described in this patent application provides a buffer management technique for speech packets over a communications network. For purposes of explanation, specific embodiments are set forth to provide a thorough understanding of the illustrative system. However, it will be understood by one skilled in the art, from reading the disclosure, that the technique may be practiced without these details. Further, although the embodiments are described in terms of a jitter buffer, it should be understood that this embodiment is illustrative and is not meant in any way to limit the practice of the disclosed system to other timing management devices. Also, the use of the terms speech packet to illustrate how the system works is not intended to infer that the illustrative system requires a specific type of audio signal. Rather, any of a variety of segmented communications may be employed in practicing the technique described herein. Moreover, well-known elements, devices, process steps, and the like, are not set forth in detail in order to avoid obscuring the disclosed system.

A typical structure and operation mode of speech communication using a packet network is depicted in FIG. 1. Speech terminals **110** and **120** are connected to the packet network **100**, each transmitting speech packets to the network and receiving speech packets from the network. It should be noted that each or any speech terminal can be combined with a data and/or visual terminal (not shown). Also, several speech terminals can be connected simultaneously to each other by the network, in what is commonly called a "conference call."

The structure of each speech terminal is given in FIG. 2. An audio input is introduced into the system as an input to the transmitting speech terminal **202**. An analog to digital (A/D) converter **200** receives the audio input as an analog signal, specifically an audio waveform. The A/D converter **200** converts the analog speech signal into a sampled and digital form, suitable for digital signal processing. The A/D converter **200** is well-known in the industry and conversion of an analog signal into digital form may be done in any number of ways understood by persons skilled in the art, such as discrete sampling. The digital signal is then forwarded to a speech encoder **210**. The speech encoder **210** further digitizes and encodes the signal with the appropriate number of bits according to speech compression algorithms, which are also well-known in the industry. The speech encoder **210** may be used through a variety of encoder/decoder (codec) standards in the industry, for example, the G.7xx codec series as specified by the International Telecommunications Union. Finally, a bit packetizing unit **220** receives the digitized audio signal and packs the bits in packets of a predetermined size, which we term Coded Speech Packages (CSPs). Additional handling or manipulation of the packets, not shown in this diagram, can include protection, encryption, and concatenation with traffic information headers, such as destination address.

The packet is then transmitted across the packet network to a receiving speech terminal **204**. Prior to the packet's receipt by the receiving speech terminal **204**, the transmitted packet is routed over various transmission paths within the packet network **100** (FIG. 1). Depending on the particular transmission route chosen and the network traffic condition, significant delay may occur between sequential packets transmitted from the transmitting speech terminal **202**. Specifically, because each packet may have traveled along a different route, one packet may travel faster or slower than another packet. In addition, some packets may have been dropped altogether to ease system congestion and will need to be transmitted again by the transmitting speech terminal **202**. Other delays may occur as a result of hardware either within the transmitting speech terminal **202** or other hardware within the packet network **100**, such as nodes of routers.

The CSPs are received from the packet network **100** at the receiving speech terminal **204**, which includes a stripping unit **250**, a jitter buffer **260**, a buffer management unit **270**, a speech decoder **240**, and a digital to analog D/A converter **230**. It is a characteristic of some packet networks to include routing information including control address and data information within each packet. The stripping unit **250** removes the control and address information to facilitate the subsequent conversion by first the speech decoder **240** and ultimately the D/A converter **230**. The jitter buffer **260** acts as an intermediate buffer at the receiver end, allowing the packets to be played out of the jitter buffer **260** at a regular or standard predetermined replay rate by other hardware in the receiving speech terminal **204** independent of the rate of arrival of the packets. Specifically, the jitter buffer **260** stores incoming speech packets before the packets are replayed. The stored packets can then be played out of the jitter buffer **260** at the regular predetermined replay rate without transferring packet data during the irregular arrival times between sequential speech packets. A regular operation mode of the speed decoder would be to decode one CSP into a single speech segment of a predetermined length, for example, 20 ms.

According to an embodiment of the present invention, the speech decoder **240** includes compression logic **264**, expansion logic **262** and a fast/slow play unit **280**. When the fast playback is enabled, the compression logic **264** compresses multiple speech packets into a reduced number of speech segments by the speech decoder **240**. When slow playback is enabled, the expansion logic **262** expands at least one speech packet into an increased number of speech segments by the speech decoder **240**. Compression is initiated upon assertion of the fast signal **272** from the buffer management unit **270** when the overflow signal **266** indicates a overflow condition exists in the jitter buffer **260**. Expansion is initiated upon deassertion of the slow signal **274** from the buffer

management unit **270** when the underflow signal **267** indicates a underflow condition exists in the jitter buffer. Compression and expansion of stored speech packets is more fully discussed in connection with FIGS. **3** and **4**.

From the jitter buffer **260**, the stored CSPs are released according to the playback rate signals **268** and **269** to the decoder **240**. The speech decoder **240** then decodes the bit information further into digital form suitable for conversion by the D/A converter **230**. Finally, the D/A converter **230** converts the digitized speech signal into an analog signal for playback by the playback unit **232** that is representative of the audio input that began the process at the transmitting speech terminal **202**.

According to a disclosed embodiment, the buffer management unit **270** monitors the contents of the jitter buffer **260**. In addition, the buffer management unit **270** sends control signals to the fast/slow play unit **280** to control the flow or transfer rate of CSPs released out of the jitter buffer **260** and the decode rate of packets from the jitter buffer **260**. Depending upon the capacity of the jitter buffer **260**, the buffer management unit **270** enables either a fast playback or a slow playback in the fast/slow play unit **280**. Specifically, when the jitter buffer **260** is relatively full, fast play is enabled. When the jitter buffer **260** is relatively empty, slow play is enable. When fast playback is enabled for packets out of the jitter buffer **260**, indicated by asserting the overflow signal **266**, the buffer management unit **270** provides a fast-play signal to the decoder **240** via the fast/slow play unit **280** and the fast playback rate signal **268** is asserted. When slow playback is enabled for packets out of the jitter buffer **260**, indicated by asserting the underflow signal **267**, the buffer management unit **270** provides a slow-play signal to the decoder **240** via the fast/slow play unit **280** and the slow playback rate signal **269** is asserted.

It should be noted that although the above described units are illustrated as separate units for exemplary purposes, it should be understood that some units might be combined in alternative embodiments. For example, the buffer management unit **270** and the fast/slow play unit **280** can be integrated without departing from the disclosed invention. Likewise, the compression logic **264** and the expansion logic **262** can be separated from the decoder unit **240** without departing from the disclosed invention.

Turning now to FIG. **3**, shown is a more detailed block diagram of the jitter buffer **260**. The size of the jitter buffer **260** can be any size permissible by the specific communications within the packet network **100**. Because the delay introduced by the jitter buffer **360** is directly proportional to its size, it is preferable to minimize the size of the jitter buffer **260**, while meeting the design considerations that will allow any irregularity in transmitted CSPs to be accounted for by the jitter buffer **260**. Each location in the jitter buffer **300** holds a CSP. A pointer **340** points to the CSP that is to be decoded and played next. The jitter buffer locations to the left of the pointer **340** hold CSPs that have already been played (and in that sense, these locations can be considered to be empty). The jitter buffer locations to the right of the pointer **340** hold CSPs that have not yet been played. There can be any number of locations between the N (Normal) location **320** and the F (Fast) location **330** and between the N location **320** and the S (Slow) location **310**. When a CSP has been decoded and played, the pointer **340** is moved one location to the right. When a new CSP is received from the network **100**, the new CSP is pushed into the jitter buffer **260** from the right. All of the unplayed CSPs are shifted one location to the left, and the pointer **340** is also moved one location to left. Note, that although the pointer **340** is positioned on the N location **320** in FIG. **3**, it can actually point to any location in the jitter buffer **300**.

The rate of the CSP decoding and playing is constant at a predetermined standard playback rate. If the rate at which the CSPs arrive from the packet network **100** is the same as the predetermined playback rate at which the CSPs are decoded and played, the pointer **340** remains at the N (Normal) location **320**, or one location to the left or to the right. However, if the temporary rate of CSP arrival from the packet network **100** is higher than the predetermined replay rate of CSP decoding and playing, more CSPs will be added to the jitter buffer **260**, the pointer **340** is shifted to the left and the overflow signal **266** (FIG. **2**) is asserted. On the other hand, if the temporary rate at which the CSPs arrive from the network **100** is lower than the predetermined playback rate at which the CSPs are decoded and played, more CSPs will be taken out of the jitter buffer **260**, the pointer **340** is shifted to the right and the underflow signal **267** is asserted.

According to another embodiment of the present invention, an overflow or underflow condition only occurs when the pointer **340** reaches a predetermined high or low level threshold of the jitter buffer **260**. Specifically,the overflow signal **266** is asserted only when the pointer **340** is moved passed a predetermined high level threshold of the jitter buffer **260**. The predetermined high level threshold represents a rate of incoming packets received by the jitter buffer **260** that exceeds the standard playback rate by a certain high threshold rate. Likewise, the underflow signal **267** is asserted only when the pointer **340** is moved passed a predetermined low level threshold of the jitter buffer **260**. The predetermined low level threshold represents a rate of incoming packets received by the jitter buffer **260** that is lower than the standard replay rate by a certain low threshold rate. Thus, slight changes in the rate of receipt of incoming packets will not trigger the disclosed fast or slow play manipulation.

Without a buffer management scheme, if the jitter in the time of arrival of the CSPs from the network exceeds a certain level, a jitter buffer can overflow or underflow. An overflow danger is detected when the pointer **340** approaches the F location **330**, and an underflow danger is detected when the pointer **340** approaches the S location **310**. According to a disclosed embodiment, the overflow indicator from the pointer **340** is used to signal a compression function for merging a number of stored speech packets into a smaller number of speech segments by the speech decoder **240**. Such a compression function is described more fully in commonly assigned U.S. Pat. No. 5,694,521 for variable speed playback of digital storage retrieval systems. Specifically, as the number of CSPs stored in the jitter buffer **260** approaches the full capacity of the jitter buffer **260**, the buffer management unit **270** will detect an overflow indicator from the pointer **340** over the overflow signal **266**. The buffer management unit **270** will initiate a compression function in the speech decoder **240** where a predetermined number of speech segments are compressed into a reduced number of speech segments. The simplest merging procedure will be the merging of two CSPs into a single speech segment, but it is also possible, for example, to merge three CSPs into two or one segments, or any other number of combination. For example, a CSP each represent a decoded speech segment of 20 ms. For a compression operation, the compression logic together **264** with the speech decoder **240** combines two CSPs to produce a speech segment of a size of 20 ms. Thus, fast playback is performed by merging a number of speech segments represented by a number of speech packets into a smaller number of speech segments

while keeping the original short-term spectrum and pitch. However, it should be understood that different combinations of spectrum and pitch can be achieved with minor modifications of the disclosed embodiment.

In addition, an underflow indicator from the pointer **340** is used to signal an expansion function for expanding a number of speech segments represented by a number of speech packets into a larger number of speech segments. Such an expansion function is described more fully in commonly assigned U.S. Pat. No. 5,694,521 for variable speed playback of digital storage retrieval systems. Specifically, as the number of CSPs stored in the jitter buffer **260** approaches the empty capacity of the jitter buffer **260**, the buffer management unit **270** will detect an underflow indicator from the pointer **340** over the underflow signal **267**. A number of speech segments represented by a number of CSPs are then expanded resulting in an increased number of speech segments. Slow playback is performed by expanding a number of CSPs into a larger number of segments, while keeping the original short-term spectrum, pitch, or other basic speech features.

From the jitter buffer **260** perspective, fast playback can be viewed as an increase in the rate of outgoing packets, and slow playback can be viewed as a decrease in the rate of outgoing packets. Fast play from the jitter buffer **260** is initiated by asserting of the fast playback rate signal **268**, while slow pay is initiated by asserting the slow playback rate signal **269**. In both cases, the speech manipulation can be performed for active and non-active speech. Fast play of the speech will increase the rate in which the CSPs are played out of the jitter buffer **260**. Fast play results in compression of speech segments into a reduced number of speech segments such that an outgoing speech segment from the decoder **240** is a single compressed version of multiple speech segments. Therefore, because multiple speech segments represented by the received CSPs are contained in the compressed outgoing speech segments, the rate of exiting CSPs will exceed the rate of incoming CSPs. Alternatively, slow play will reduce the rate in which the CSPs are played out of the jitter buffer **260**. Slow play results in expansion of speech segments into an increased number of speech segments such that an outgoing speech segment from the decoder **240** is an expanded version of only a portion of a speech segment. Therefore, because only a portion of a speech segment represented by an incoming CSP is contained in the expanded outgoing speech segment, the rate of exiting CSPs will be lower than the rate of incoming CSPs.

If there is no jitter in the time of arrival of the packets from the network **100**, the jitter buffer **260**, the buffer management unit **270**, and the fast/slow play unit **280** operate to pass the audio signal through the decoder path in a reverse manner to the encoder path. No compression or expansion is performed. The CSPs are then stripped to the bits. The bits are decoded to generate the sampled and digitized speech, which is then converted into an analog signal by the D/A converter.

Turning now to FIG. **4***a*, illustrated is an exemplary timing relationship between sequential speech packets received from the packet network **100** (FIG. **1**). The top set of packets represents the jitter buffer input at location ① as shown in FIG. **2**. Because of various delays within the transmitting speech terminal **202** and/or various delays within the packet network **100**, the stream of transmitted packets is received by the jitter buffer **260** in an asynchronous manner. Specifically, the packets **P3**, **P4**, **P9**, **P10** and **P11** arrive at the right time, while **P5**, **P6**, **P7** and **P8** arrive late. Note the sparse arrival time of **P5** and **P6**, which is compensated by the dense arrival time of **P7** and **P8**.

According to a disclosed embodiment, a normal event occurs where the time of arrival for incoming packets to the jitter buffer **260** is approximately equal to the predetermined standard replay rate for subsequent decoding and converting of the audio signal. A fast arrival event occurs when the rate of arrival of packets into the jitter buffer **260** is significantly higher than the predetermined replay rate for subsequent decoding and converting of the audio signal into an audible output. Finally, a slow event occurs when the rate of arrival between packets into the jitter buffer **260** is significantly lower than the predetermined replay rate for subsequent decoding and converting of the packets into an audible output. According to another embodiment of the present invention, a fast or slow event occurs only when the incoming rate of received packets exceeds a high threshold rate corresponding to a high threshold level in the jitter buffer **260** or is lower than a low threshold rate corresponding to a low threshold level in the jitter buffer **260**, respectively.

The middle packet stream represents the output of packets from the jitter buffer **260** at location ② shown on FIG. **2**. Since **P5** does not arrive at time t+3, a slow event at time t+3 occurs. The buffer management unit **270** signals the speech decoder **240** of the slow event by asserting the slow signal **274**. Expansion logic **262** in the speech decoder **240** expands the **P3** speech packet such that subsequent decoding results in speech packets S3A and S3B over two output speech segments. Speech segments S3A and S3B are the decoded speech signal information represented by the pre-decoded speech packet **P3**. **P6** and **P7** arrive late, but since **P3** was already expanded, the buffer is not empty and **P4** and **P5** are played at a normal rate. Since **P8** now arrives before **P6** is played, **P6** and **P7** are played out of the jitter buffer **260** in a fast play mode during time t+7. Upon a fast event at time t+6 and t+7, the buffer management unit **270** signals the speech decoder **240** of the fast event by asserting the fast signal **272**. Compression logic **264** in the speech decoder **240** compresses the **P6** and **P7** speech packets such that subsequent decoding results in speech packet S6+7. Speech packet S6+7 is the decoded speech signal information represented by both the pre-decoded speech packets **P6** and **P7**.

As described above, although a 2:1 fast play mode is shown for exemplary purposes, any ratio of fast play may occur where the outgoing CSP from the jitter buffer **260** consists of more than one of the CSPs stored within the jitter buffer **260**. The slow arrival event at time t+3 results in a slow play mode at times t+3 and t+4. Specifically, the packets received by the jitter buffer **260** are output at a slower rate than the predetermined replay rate. Here again, although a 1:2 slow play mode is shown for exemplary purposes, any ratio of slow play may be used.

Finally, the bottom stream of speech segments illustrates the timing for subsequent decoding and converting of the speech packets into corresponding speech segments, at location ③ shown in FIG. **2**. The consistent time of arrival interval of the bottom stream of speech segments may be any predetermined time interval, 20 ms for example. It is this regular and consistent rate of arrival on which smooth and continuous audible output relies.

It is important to note that the compression and expansion operations are performed on speech packets output from the jitter buffer **260** at a time when the arrival of speech packets into the jitter buffer **260** signals such operation. Therefore, since the output of the jitter buffer **260** is delayed from the input, the compression and expansion operations are not necessarily performed on the speech packets, or the speech segments represented by the speech packets, that actually cause the signaling of either the fast or slow play mode.

Turning to FIG. 4*b*, another example is illustrated where the rate of arrival of speech packets results in either normal, compressed or expanded decoding into speech segments. Since a fast event occurs from an accelerated arrival of packets at time t+3, the packets in the jitter buffer **260** are played out at a faster rate such that P3 and P4 are played in a single segment. From this output the compression logic **264** is initiated allowing the decoder **240** to output a single compressed speech segment containing speech information represented by both P3 and P4. Similarly, the slow arrival at time t+6 results in expanded speech segments S7A and S7B over two speech segments.

The fast or slow play can be performed for all speech segments, both silent and active. In this way immediate and continuous jitter buffer manipulation is achieved without removing speech segments or inserting artificially generated speech segments. It is also possible to restrict jitter buffer manipulation to stationary voiced, stationary unvoiced, and inactive speech segments, and to avoid jitter buffer manipulation during the non-stationary portions of the speech, such as transitions. With this approach, it is estimated that more than 90% of the speech segments can be manipulated without audible speech quality degradation. By avoiding the buffer correction during transition speech, where the fast/slow playback can introduce some distortion, the speech quality is increased while still able to perform an efficient buffer manipulation.

According to an alternate embodiment, a buffer management scheme is provided with several degrees of overflow and underflow danger. As the pointer **340** starts to move to the left or to the right of the jitter buffer **260**, the level of danger can be increased. According to the level of overflow/underflow danger, the urgency in the need for buffer manipulation is increased, and accordingly, the level of manipulation. For example, on a low level of overflow urgency, the fast play will only combine 3 segments of speech into 2 segments (3:2 faster ratio) and will operate only during stationary speech, stationary unvoiced, or inactive speech segments. As the level of overflow urgency increases, for example, the fast play can start to combine 2 segments into a single segment (2:1 faster ratio) and can perform the speech manipulation for all segments, regardless of their nature.

Therefore according to a disclosed embodiment, continuous play of asynchronously transmitted speech packets is provided through manipulation of data packets within a jitter buffer. An overflow indicator signals the receiving terminal to accelerate the rate of play of outgoing packets from the jitter buffer. Playback is accelerated by compressing a predetermined number of speech packets into a reduced number of speech segment. Alternatively, an underflow indicator instructs the receiving terminal to decelerate playing of outgoing speech packets from the jitter buffer. Deceleration is achieved by expanding a predetermined number of speech packets within the jitter buffer into an increased number of speech segment in the decoder output. Subsequent decoding of the packets from the jitter buffer is performed according to a fast or slow play status corresponding to the packet to be decoded. Specifically, compressed speech packets are decoded according to a fast decode algorithm while expanded speech packets are decoded according to a slow decode algorithm. In this way, delay resulting from asynchronous time of arrival between sequential speech packets is avoided by providing outgoing speech packets from the jitter buffer at a suitable rate. In addition, jitter buffer management is achieved without removing portions of the transmitted packets or by adding artificially generated packets to the sequence of the packets in the jitter buffer. The disclosed jitter buffer management techniques address many of the concerns associated with jitter buffers.

The foregoing disclosure and description of the various embodiments are illustrative and explanatory thereof, and various changes in communication network, the descriptions of the jitter buffer, the receiver, and other circuitry, the organization of the components, and the order and timing of steps taken, as well as in the details of the illustrated system may be made without departing from the spirit of the invention.

I claim:

1. A method of controlling playback of audio signals over a communication network, the method comprising:

receiving a plurality of audio packets;

storing temporarily the plurality of audio packets;

executing playback of the plurality of audio packets;

compressing the plurality of audio packets to accelerate the playback of the plurality of audio packets when a rate of receipt of audio packets is greater than a predetermined upper replay rate; and

decompressing the plurality of audio packets to decelerate the playback of the plurality of audio packets when the rate of receipt of the plurality of audio packets is less than a predetermined lower replay rate.

2. The method of claim **1**, further comprising:

decoding the plurality of audio packets.

3. The method of claim **1**, the accelerating step further comprising:

compressing an audio packet.

4. The method of claim **3**, wherein the compressing step reduces the number of the plurality of audio packets.

5. The method of claim **1**, the accelerating step further comprising:

compressing a speech segment represented by an audio packet.

6. The method of claim **1**, the decelerating step further comprising:

expanding an audio packet.

7. The method of claim **6**, wherein the expanding step increases the number of the plurality of audio packets.

8. The method of claim **1**, the decelerating step further comprising:

expanding a speech segment represented by an audio packet.

9. The method of claim **1**, further comprising the step of:

detecting the rate of receipt of the plurality of audio packets.

10. The method of claim **9**, the plurality of audio packets being stored in a jitter buffer, detecting step comprising the step of:

determining a location of a jitter buffer using an address pointer of the jitter buffer.

11. The method of claim **10**, wherein the jitter buffer address pointer points to an address of the jitter buffer corresponding to a relatively full level of the jitter buffer when the rate of receipt of the audio packets is higher than the predetermined replay rate and the jitter buffer address pointer points to an address of the jitter buffer corresponding to a relatively empty level of the jitter buffer when the rate of receipt of the audio packets is lower than the predetermined replay rate.

12. A receiver configured for continuous playback of audio packets, the receiver comprising:

a jitter buffer to store a plurality of audio packets;

a jitter buffer controller coupled to the jitter buffer to monitor capacity of the jitter buffer, the jitter buffer controller accelerating playback of the plurality of audio packets out of the jitter buffer when a rate of receipt of the plurality of audio packets is greater than a predetermined upper replay rate and decelerating the playback of the plurality of audio packets out of the jitter buffer when a rate of receipt of the plurality of audio packets is lower than a predetermined lower replay rate; and

a decoder to decode the stored audio packets, the decoder compressing an audio packet when a rate of receipt of the plurality of audio packets is greater than a predetermined upper replay rate, the decoder expanding an audio packet when the rate of receipt of the plurality of audio packets is lower than the predetermined lower replay rate.

13. The receiver of claim 12, wherein the jitter buffer controller provides a fast play signal to the decoder during accelerated playback and provides a slow play signal to the decoder during decelerated playback.

14. The receiver of claim 12, wherein the jitter buffer provides an overflow indicator signal to the buffer controller to initiate accelerated playback and the jitter buffer provides an underflow indicator signal to initiate decelerated playback.

15. The receiver of claim 12, the decoder compressing an audio packet when a rate of receipt of the plurality of audio packets is greater than a predetermined upper replay rate, the decoder expanding an audio packet when the rate of receipt of the plurality of audio packets is lower than the predetermined lower replay rate.

16. The receiver of claim 12, wherein a compressed audio packet is decoded according to a corresponding compression decode algorithm and an expanded audio packet is decoded according to a corresponding expansion decode algorithm.

17. A communications network configured for continuous playback of asynchronously transmitted audio packets, comprising:

a transmitter to transmit an audio packet;

a receiver to receive an audio packet, comprising:

a jitter buffer for storing received audio packets;

a jitter buffer controller coupled to the jitter buffer to monitor capacity of the jitter buffer, the jitter buffer controller accelerating playback of the plurality of audio packets out of the jitter buffer when a rate of receipt of the plurality of audio packets is greater than a predetermined upper replay rate and decelerating the playback of the plurality of audio packets out of the jitter buffer when a rate of receipt of the plurality of audio packets less than a predetermined lower replay rate;

a decoder to decode the stored audio packets, the decoder compressing a speech segment represented by an audio packet when a rate of receipt of the plurality of audio packets is greater than a predetermined upper replay rate, the decoder expanding a speech segment represented by an audio packet when the rate of receipt of the plurality of audio packets is lower than the predetermined lower replay rate;

a converter for converting the audio packets into an audible signal; and

a playback device for replaying the audible signal at the predetermined replay rate.

18. The communications network of claim 17, wherein the jitter buffer provides an overflow indicator signal to the buffer controller to initiate accelerated playback and the jitter buffer provides an underflow indicator signal to initiate decelerated playback.

19. The communications network of claim 17, wherein the jitter buffer controller provides a fast play signal to the decoder during accelerated playback and provides a slow play signal tot the decoder during decelerated playback.

20. The communications network of claim 17, wherein a compressed speech segment is decoded according to a corresponding compression decode algorithm and an expanded speech segment is decoded according to a corresponding expansion decode algorithm.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,377,931 B1
DATED : April 23, 2002
INVENTOR(S) : Eyal Shlomot

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 11,
Lines 31-36, pleae delete claim 15.

Column 11, lines 37-43 through Column 12, lines 1-41,
Renumber claims 16-20 as claims 5-19.

Signed and Sealed this

Sixth Day of May, 2003

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006658027B1

(12) **United States Patent**
Kramer et al.

(10) **Patent No.: US 6,658,027 B1**
(45) **Date of Patent: Dec. 2, 2003**

(54) **JITTER BUFFER MANAGEMENT**

(75) Inventors: **Kris W. Kramer**, Kanata (CA); **Chris C. Forrester**, Ottawa (CA); **Robert Joly**, Nepean (CA)

(73) Assignee: **Nortel Networks Limited** (CA)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/374,528**

(22) Filed: **Aug. 16, 1999**

(51) **Int. Cl.**[7] **H04J 3/06**

(52) **U.S. Cl.** **370/516**; 370/517

(58) **Field of Search** 370/229, 232, 370/235, 242, 252, 253, 516, 517, 519

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,389,032	B1 *	5/2002	Cohen	370/412
6,434,606	B1 *	8/2002	Borella et al.	709/214
6,452,950	B1 *	9/2002	Ohlsson et al.	370/516

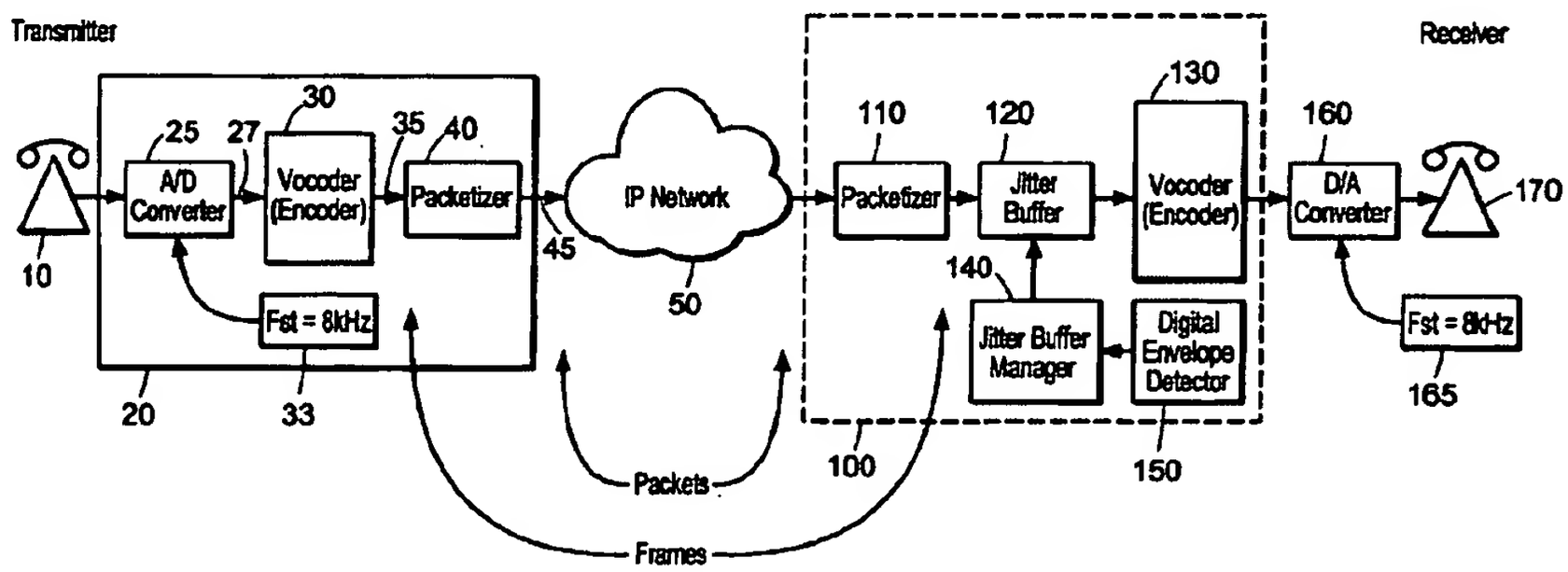
* cited by examiner

Primary Examiner—Wellington Chin
Assistant Examiner—Brenda Pham
(74) *Attorney, Agent, or Firm*—Steubing McGuinness & Manaras LLP

(57) **ABSTRACT**

In order to compensate for rate mismatches between near end (receiving) and far end (transmitting) devices, intelligent jitter buffer management is implemented by apparatus comprising: a data interface for receiving frames from a data network; a jitter buffer for temporarily storing said frames; a detector for detecting frames which satisfy a criteria; and a buffer manager for controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria. The criteria can include silence frames or frames received with errors. The condition can include a high water mark (high threshold), and a low water mark (low threshold). If the far end transmitter transmits at a faster rate than the near end receiver, the jitter buffer will eventually become full beyond the high water mark, in which case frame(s) which satisfy the criteria will be deleted. If the far end transmitter transmits at a slower rate than the near end receiver, the jitter buffer will eventually become depleted below the low water mark, in which case silence frame(s) will be inserted after received silence frames.

20 Claims, 6 Drawing Sheets



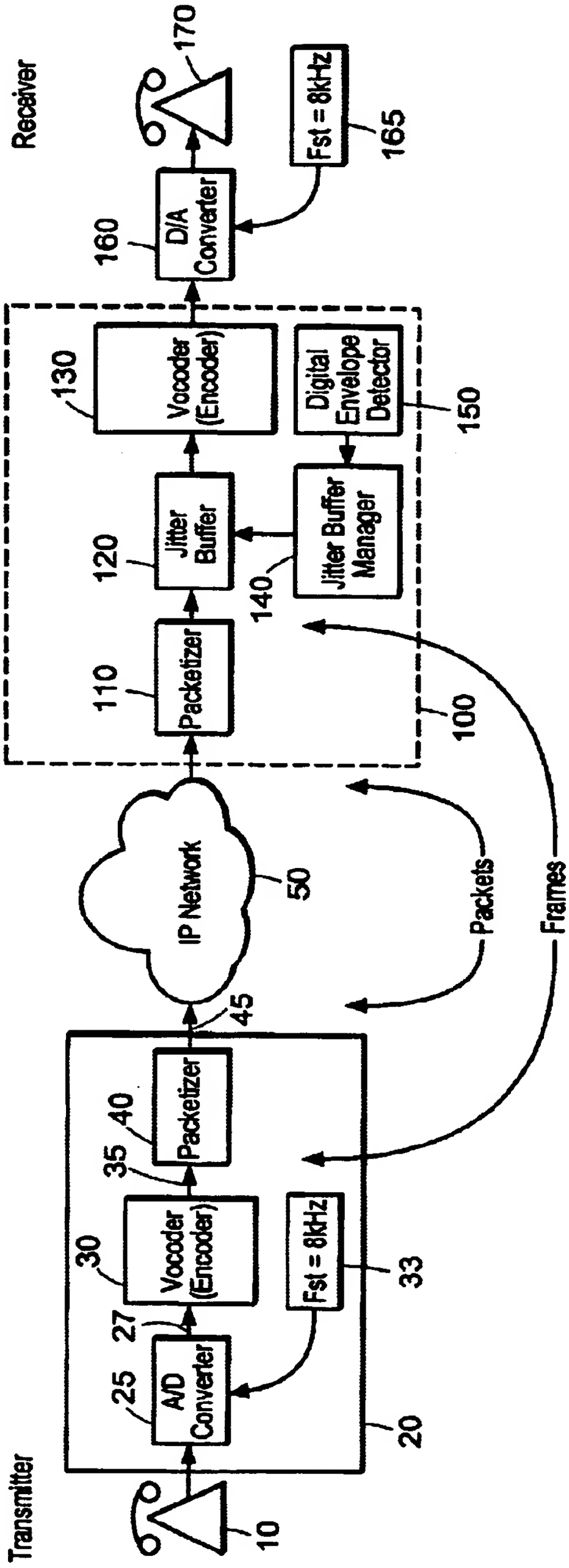

Receiver
170
Fs1 = 8kHz
165
160
D/A Converter
130
Vocoder (Encoder)
Digital Envelope Detector
150
120
Jitter Buffer
Jitter Buffer Manager
140
110
Packetizer
100
IP Network
50
45
Packets
Frames
35
40
Packetizer
30
Vocoder (Encoder)
Fs1 = 8kHz
33
27
25
A/D Converter
20
Transmitter
10

FIG. 1

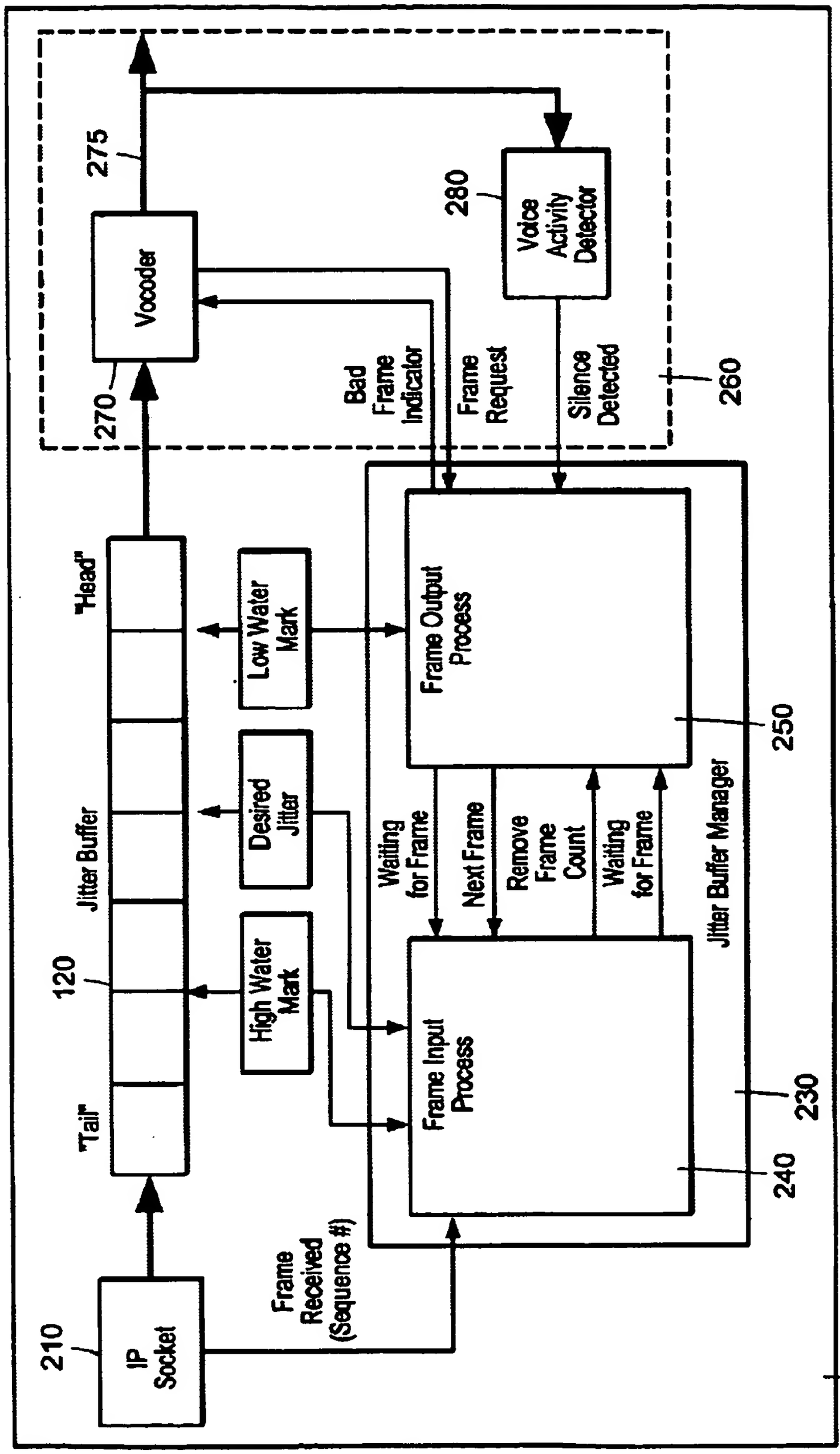

275
Vocoder
280
Voice Activity Detector
270
Bad Frame Indicator
Frame Request
Silence Detected
260
"Head"
Jitter Buffer
"Tail"
120
Low Water Mark
Desired Jitter
High Water Mark
Frame Output Process
Waiting for Frame
Next Frame
Remove Frame Count
Waiting for Frame
Jitter Buffer Manager
250
Frame Input Process
230
240
210
IP Socket
Frame Received (Sequence #)
100

FIG. 2

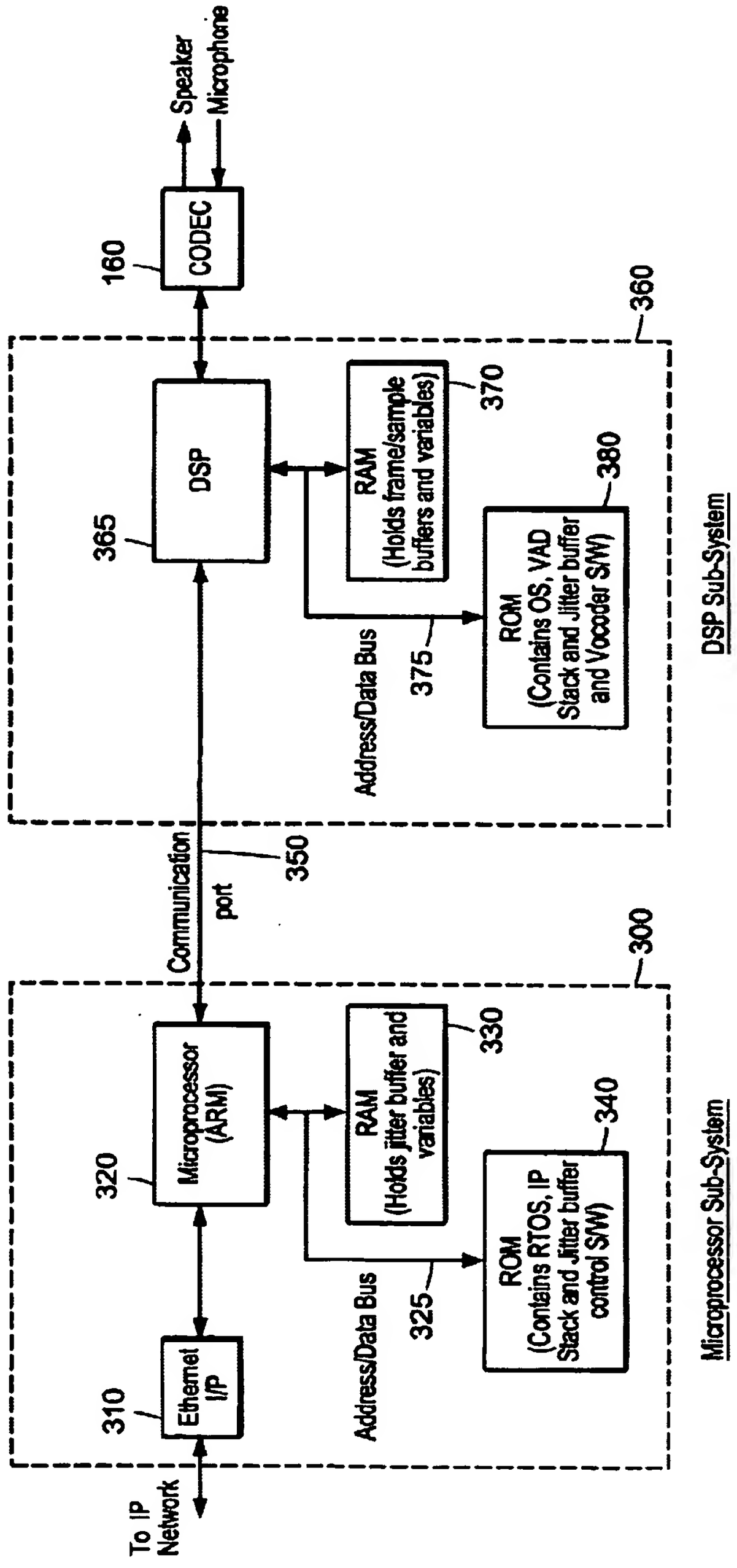

Speaker
Microphone
160
CODEC
360
DSP
365
RAM
(Holds frame/sample buffers and variables)
370
ROM
(Contains OS, VAD Stack and Jitter buffer and Vocoder S/W)
380
Address/Data Bus
375
DSP Sub-System
Communication port
350
300
Microprocessor (ARM)
320
RAM
(Holds jitter buffer and variables)
330
ROM
(Contains RTOS, IP Stack and Jitter buffer control S/W)
340
Address/Data Bus
325
Ethernet I/P
310
To IP Network
Microprocessor Sub-System

FIG. 3

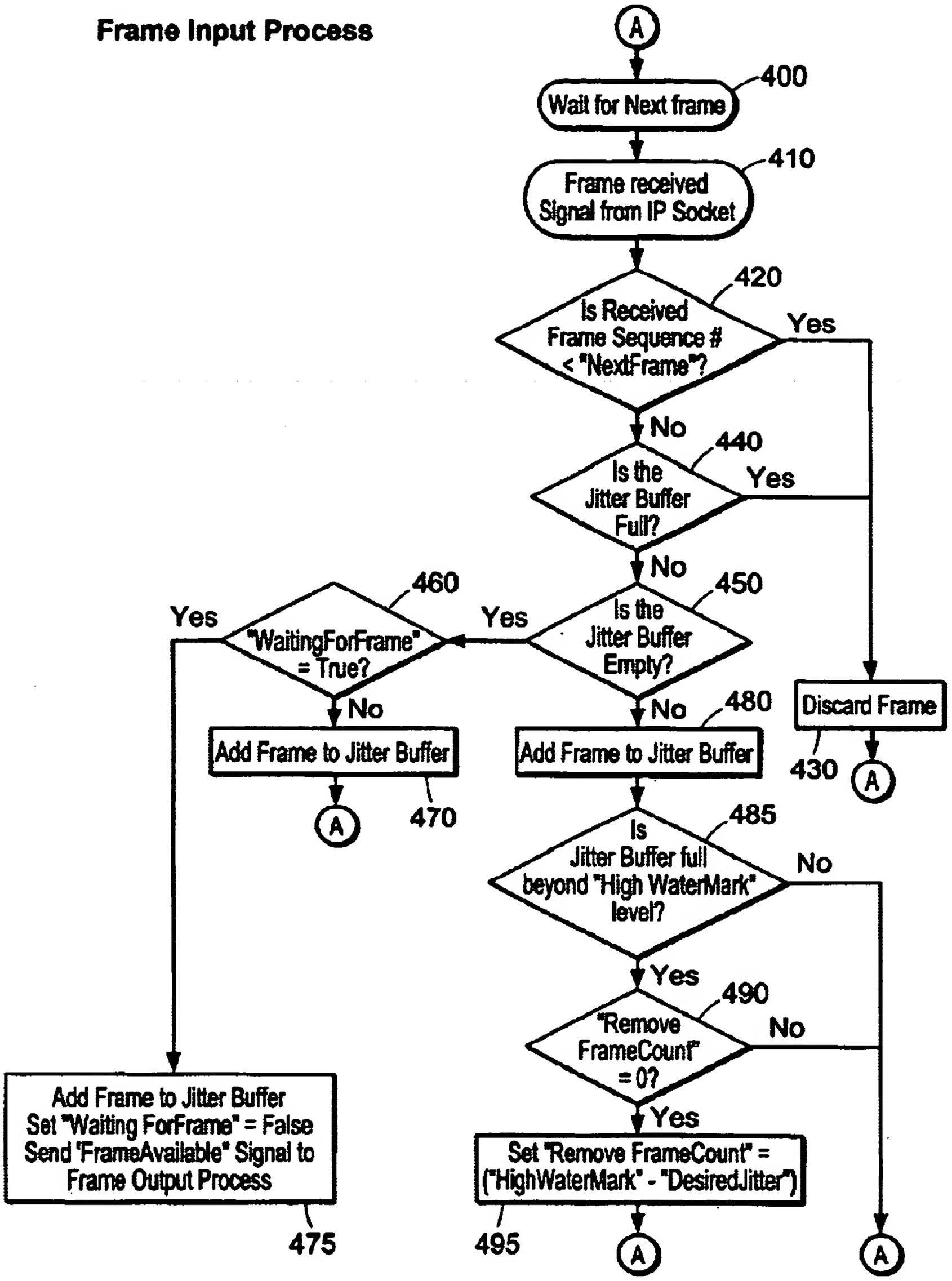

Frame Input Process
A
Wait for Next frame
400
Frame received Signal from IP Socket
410
Is Received Frame Sequence # < "NextFrame"?
420
Yes
No
Is the Jitter Buffer Full?
440
Yes
No
Is the Jitter Buffer Empty?
450
Yes
"WaitingForFrame" = True?
460
Yes
No
Add Frame to Jitter Buffer
470
A
No
Add Frame to Jitter Buffer
480
Discard Frame
430
A
Is Jitter Buffer full beyond "High WaterMark" level?
485
No
Yes
"Remove FrameCount" = 0?
490
No
Yes
Set "Remove FrameCount" = ("HighWaterMark" - "DesiredJitter")
495
A
A
Add Frame to Jitter Buffer
Set "Waiting ForFrame" = False
Send 'FrameAvailable" Signal to Frame Output Process
475

FIG. 4

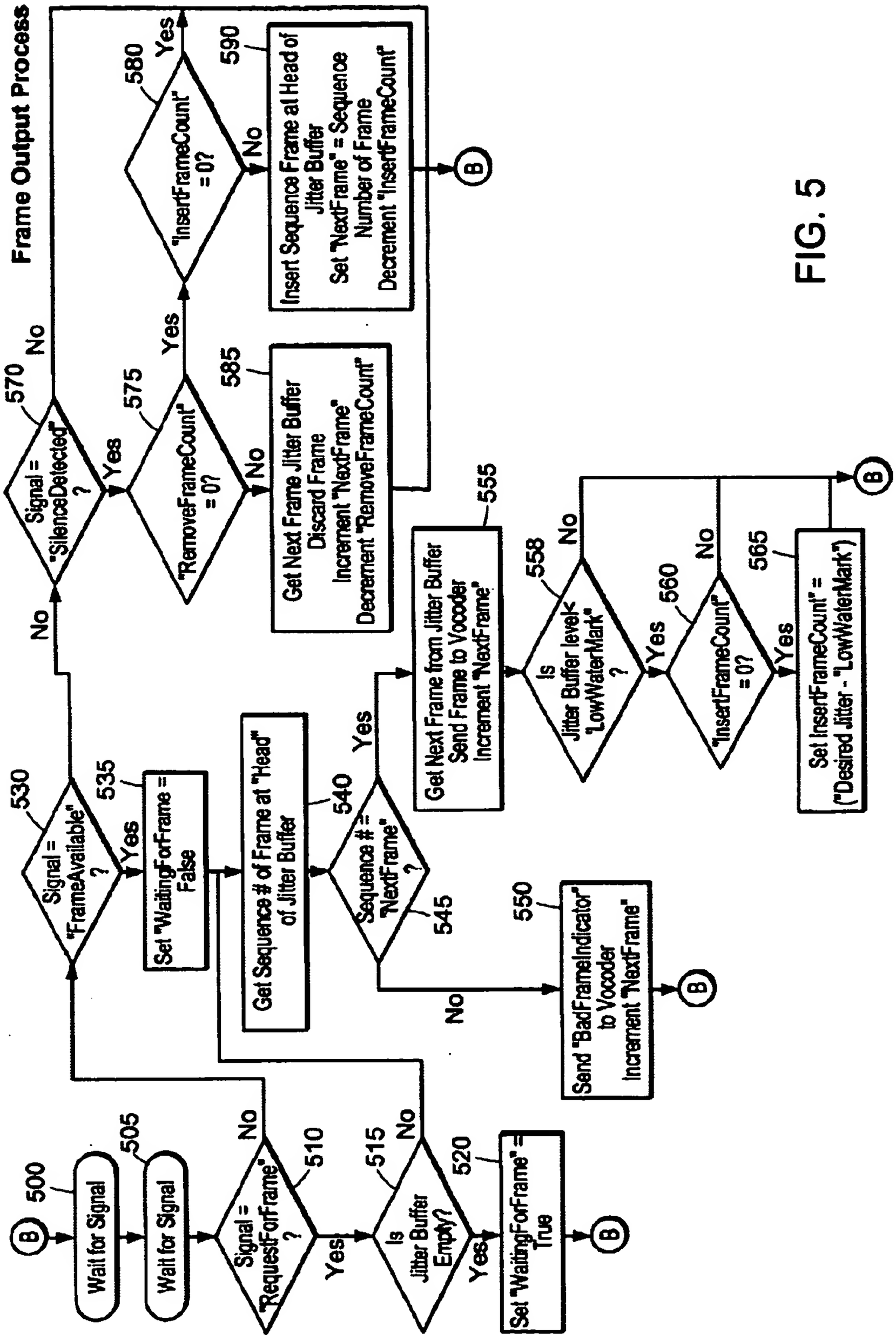

Frame Output Process
500
505
510
515
520
530
535
540
545
550
555
558
560
565
570
575
580
585
590
Wait for Signal
Wait for Signal
Signal = "RequestForFrame" ?
Is Jitter Buffer Empty?
Set "WaitingForFrame" = True
Signal = "FrameAvailable" ?
Set "WaitingForFrame" = False
Get Sequence # of Frame at "Head" of Jitter Buffer
Sequence # = "NextFrame" ?
Get Next Frame from Jitter Buffer Send Frame to Vocoder Increment "NextFrame"
Is Jitter Buffer level< "LowWaterMark" ?
"InsertFrameCount" = 0?
Set InsertFrameCount" = ("Desired Jitter - "LowWaterMark")
Send "BadFrameIndicator" to Vocoder Increment "NextFrame"
Signal = "SilenceDetected" ?
"RemoveFrameCount" = 0?
Get Next Frame Jitter Buffer Discard Frame Increment "NextFrame" Decrement "RemoveFrameCount"
"InsertFrameCount" = 0?
Insert Sequence Frame al Head of Jitter Buffer Set "NextFrame" = Sequence Number of Frame Decrement "InsertFrameCount"
Yes
No
B

FIG. 5

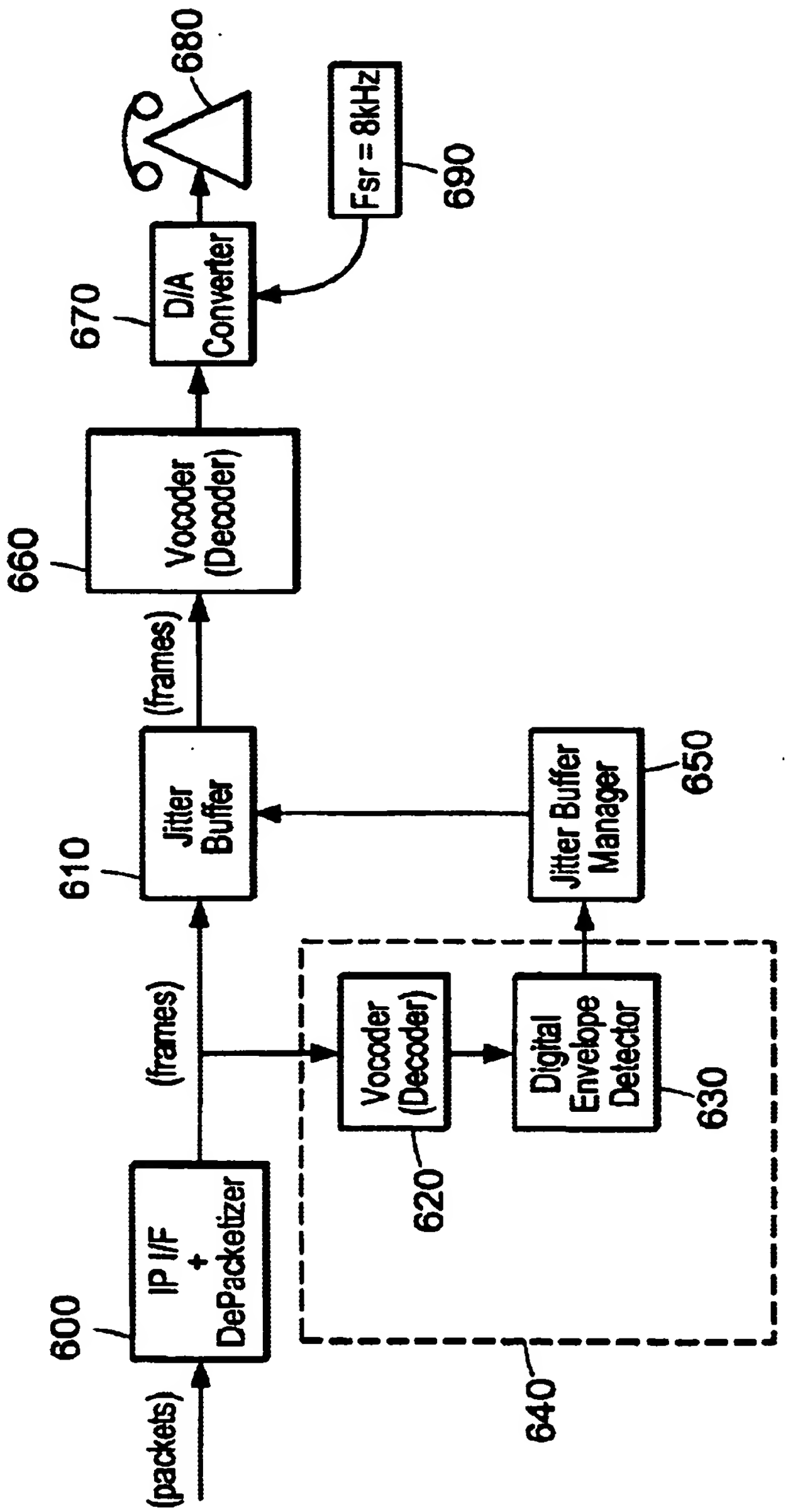

680
Fsr = 8kHz
690
670
D/A Converter
660
Vocoder (Decoder)
(frames)
610
Jitter Buffer
Jitter Buffer Manager
650
(frames)
Vocoder (Decoder)
Digital Envelope Detector
630
620
600
IP I/F + DePacketizer
(packets)
640

FIG. 6

JITTER BUFFER MANAGEMENT

FIELD OF THE INVENTION

The present invention relates to data transmission of streaming data. The invention is particularly suited for voice over packet data networks, for example Voice over Internet Protocol (VoIP) networks.

BACKGROUND TO THE INVENTION

For VoIP networks, audio signals are digitized into frames and transmitted as packets over an IP network. The transmitter sends these packets at a constant transmission rate. An appropriately configured receiver will receive the packets, extract the frames of digital data and convert the digital data into analog output using a digital to analog (D/A) converter. One of the characteristics of an IP network is that packets will not necessarily arrive at their destination at a constant rate, due to variable delays through the network. However, digital audio data (for example a digitized voice conversation) must be played out at a constant output rate in order to reconstruct the audio signal, and the D/A converter operates at such a constant output rate.

A known solution for this problem is to implement a jitter buffer in the receiver. A jitter buffer stores frames as they are received from the network. After several frames are loaded into the buffer, the frames in the buffer are output at the constant output rate. As long as the average rate of reception of the packets is equal to the constant output rate, the jitter buffer allows the packets to be output at the constant output rate even though they are not necessarily received at a constant rate.

In traditional (e.g., PSTN) digital telephony systems, end points are synchronized by a common master clock in order to ensure that the D/A and A/D converters at both ends operate at the same data rate. In other words, the PSTN is a synchronous network, and thus the constant transmission rate is the same as the constant output rate. However in a packet based system, there is no common clock to ensure synchronization of the data rates. Thus the two endpoints will typically have marginally different data rates. Thus the constant output rate from the jitter buffer will differ from the far-end constant transmission rate.

For example, let us assume that the clock rate of the A/D converter of the far-end transmitter is slightly faster than the clock rate of the D/A converter of the receiver. This will result in the far end transmitter sending digital samples of audio data at a rate faster then the local receiver will be converting the digital samples into analog. This will result in a output rate of the jitter buffer which is slower than the far-end transmission rate. Eventually this will result in the jitter buffer becoming full. In traditional jitter buffer designs, this will result in a random discard of a frame, which degrades audio quality.

Thus, while known jitter buffer techniques can compensate for variable transmission delays through the network (provided the average rate of reception is equal to the constant output rate), the jitter buffer can be either depleted or filled to capacity due to a rate mismatch between the far-end transmitter and the local receiver.

There exists a need to overcome this problem.

SUMMARY OF THE INVENTION

An object of the present invention is to provide a system which monitors the jitter buffer in order to determine conditions when a frame will need to be deleted or inserted. When such a condition exists, the system intelligently selects frames for insertion or deletion based on a criteria which reduces the impact of such an insertion/deletion. Thus, the system includes a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame.

For example, for a voice conversation, there are silence frames which result from inherent gaps in speech. A silence frame is better to delete than a frame with actual speech content. Furthermore, if a frame needs to be inserted, it is better to insert a silence frame immediately after a silence frame than between two content frames. Thus in one embodiment, the criteria includes the detection of silence, and the detector includes a silence detector (for example a Voice Activity Detector (VAD) or envelope detector) to detect silent frames. Such a system extends silence intervals, by detecting silence and inserting silence frame(s) at head of jitter buffer, when the jitter buffer is depleted below a depletion threshold (called the low water mark). Similarly when the jitter buffer is filled beyond a filled threshold (called the high water mark), the system deletes silence frames. Note that in this specification silence (or a silent frame) can include background and/or comfort noise.

Another example of criteria indicative of the impact of the insertion or deletion of a frame includes whether a frame is received with errors. Another criteria includes information associated with received frames about the mode of operation of the apparatus when the apparatus is using some kind of echo control or switched loss algorithm (for example, during handsfree operation). For example an indication that a frame is received while a terminal is transmitting (i.e. near end talking) or in quiescent mode would indicate that such a frame can be deleted with minimal audible impact.

In accordance with a broad aspect of the present invention there is provided a method of managing a jitter buffer comprising the steps of:

receiving frames from a data network;

storing received frames into said jitter buffer;

detecting frames which satisfy a criteria; and

controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria.

According to a further aspect of the invention, said condition comprises a high water mark and a low water mark and wherein said controlling step comprises:

deleting a frame from said jitter buffer when the high water mark is exceeded and when said criteria is satisfied; and

inserting a frame into said jitter buffer when said buffer is depleted below said low water mark and when said criteria is satisfied.

In accordance with another broad aspect of the present invention there is provided Apparatus comprising:

a data interface for receiving frames from a data network

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria; and

a buffer manager for controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention, together with further objects and advantages thereof will be further understood from the

following description of the preferred embodiments with reference to the drawings in which:

FIG. **1** illustrates a VoIP apparatus connected to an IP network according to an embodiment of the invention.

FIG. **2** is a functional block diagram illustrating the VoIP apparatus of FIG. **1**.

FIG. **3** is a hardware block diagram showing the elements in the VoIP apparatus which implement the functions of FIG. **2**.

FIG. **4** is a flowchart of the Frame Input Process according to an embodiment of the invention.

FIG. **5** is a flowchart of the Frame Output Process according to an embodiment of the invention.

FIG. **6** is an alternative embodiment of the invention which implements a detector before the jitter buffer in order to provide more control of the frames in the jitter buffer.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

We will describe the preferred embodiments of the invention with reference to the example of a Voice over IP (VoIP) apparatus. However the invention is not limited to IP, and can be used with other packet data networks. Furthermore, for convenience, we will discuss the examples of a VoIP apparatus which forms part of, or connects to a single voice terminal. However, it should be noted that the invention can be implemented in a network device, for example a PSTN-IP gateway, or PBX or Key system.

FIG. **1** illustrates a voice over IP apparatus connected to an IP network according to an embodiment of the invention. In FIG. **1**, an IP network **50** provides transmission of voice packets from a far-end transmitter **20** to a local receiver **100**. In this particular example, the far-end transmitter includes a handset **10** and a voice over IP (VoIP) transmitter **20**. In this example, the voice over IP transmitter **20** includes an analog to digital (AID) converter **25**, for example, a CODEC. The A/D converter **25** digitizes audio from the handset **10** at a constant transmission rate which depends on a transmit clock frequency provided by a local crystal **33**. The output from the A/D converter **25**, for example, a PCM (Pulse Code Modulated) signal **27** is sent to a frame-oriented vocoder which produces frames of digital/audio data according to a particular vocoding algorithm in use. These frames **35** are sent to an IP packetizer interface **40** which structures the frames into IP packets **45** according to known Internet protocols and then sends these packets to the IP network **50** for transmission to the receiver. The packets **45** are sent at a constant transmission rate that is dictated by the A/D converter **25** and its input clock frequency from the crystal **33**.

The IP network adds a variable delay such that the receiver does not necessarily receive the packets at the same rate as they were transmitted. The receiver VoIP apparatus **100** comprises an IP depacketizer interface **110** which receives the IP packets from the IP network **50** and includes a depacketizer for deconstructing the packets into their component frames. These frames are then stored temporarily in a jitter buffer **120** which is controlled by a jitter buffer manager **140**. The frames are then sent to a vocoder **130** which deconstructs the frames according to a particular vocoder routine. The vocoder **130** produces, for example, PCM output, which is sent to the D/A converter **160** (e.g., a CODEC) which converts the digital signal into an analog audio signal which is sent to the handset **170**. The constant output rate of the receiver **100** is dictated by the clock frequency of clock **165** which controls the "sampling" or "playback" rate of the CODEC.

As stated previously in the background section, conventional jitter buffers can typically compensate (or at least alleviate) random delays of packet transmission through the IP network **50**. However, conventional jitter buffers fail to compensate for a rate mismatch between the clock of crystal **33** in the far-end transmitter and the clock frequency of the crystal **165** in the local receiver. This rate mismatch results in a constant transmission rate that differs from the constant output rate. This will tend to either deplete or fill the jitter buffer. This typically results in the random insertion of a silence frame or the random discard of a frame. Either way, the audio quality is degraded.

One aspect of the invention reduces such audio degradation by including a detector for detecting frames which satisfy criteria so as to intelligently insert or discard frames as needed. Thus, the jitter buffer manager controls the frames stored in the jitter buffer, based on the condition of the jitter buffer, and if necessary, based on frames which satisfy said criteria. In the embodiment shown in FIG. **1**, the criteria includes a frame which represents silence and the detector comprises an envelope detector (or a voice activity detector) coupled to the output of the vocoder **130**. This detector determines whether the output from the voice coder represents silence (which, for the purposes of the specification, can include comfort noise). If this criteria is detected, a signal is sent to the jitter buffer manager indicating that the frame sent to the vocoder **130** represents a silent frame. The jitter buffer manager will use this information to control the insertion or deletion of frames in the jitter buffer as required, depending on the condition of the jitter buffer, as will be described in more detail below.

Note that the VoIP apparatus can take a variety of forms. In one form, the voice over IP apparatus forms part of an integrated phone, which includes the VoIP apparatus, a screen display, a touch pad and a handset. Alternatively, the VoIP apparatus can include an interface for allowing a digital phone (for example, a phone adapted to work with a digital key system or digital PBX) to communicate via a packet network by coupling to the VoIP apparatus. In this example, the D/A converter will be located in the digital phone, and the VoIP apparatus will include a clock signal which will determine the constant output rate. Furthermore, the phone will derive its sampling rate by phase locking to the output clock signal provided by the VoIP apparatus (as is known in the PBX art). The VoIP apparatus, in fact, can form part of a key system or PBX. Furthermore, the VoIP apparatus can include a subscriber line interface circuit (SLIC) for coupling to a conventional analog phone. Furthermore, note that the transmitter **20** can form part of a PSTN-IP Gateway, as can the VoIP apparatus **100**.

FIG. **2** is a functional block diagram of the VoIP apparatus **100** of FIG. **1**. The apparatus includes functional blocks representing an IP socket **210**, a jitter buffer **120**, a jitter buffer manager **230** and a DSP **260** including a frame criteria detector. The jitter buffer manager includes a frame input process **240** and a frame output process **250**. The jitter buffer manager will typically be implemented as software instructions executed on a controller, for example, a microprocessor or an advanced RISC machine (ARM) and associated memory.

In FIG. **2**, an IP socket **210**, which is the application programming interface (API) used to gain access to the IP network, delivers the IP packets to the frame input process **240**. In the embodiment shown, the DSP **260** includes a

vocoder **270** and a voice activity detector **280** coupled to the output of the vocoder **270** which determines whether silence is detected at the output of the vocoder. In this embodiment of the invention, there is associated with the jitter buffer **120**, a high water mark, a desired jitter level and a low water mark. Both the high water mark and the low water mark represent buffer conditions used by the jitter buffer manager in determining whether a frame or frames need to be inserted or deleted from the jitter buffer. If the jitter buffer depth drops below the low water mark threshold, this indicates that jitter in arrival rate of the received packets may propagate through the jitter buffer to the vocoder and affect the audio quality of the signal. If the jitter buffer depth expands to the high water mark, this signals that the total delay of the jitter buffer is getting too long and this can also negatively affect the perceived audio quality.

The jitter buffer is a variable length buffer usually on the order of a few tens of milliseconds long. The jitter buffer should be long enough to be able to store a sufficient number of frames such that the jitter buffer manager can accommodate the high water mark threshold, as explained below, while still allowing for some headroom for short packet bursts, over the entire range of expected desired jitter buffer depths. The jitter buffer length is also constrained by cost and performance factors and the desired jitter level. The desired jitter level represents a trade-off between added delay, which is generally undesirable, and the need to compensate for large variations in packet reception rates.

The jitter buffer manager inserts received frames at the tail of the jitter buffer and moves the frames to the head of the buffer while maintaining proper ordering. Frames propagate to the head of the jitter buffer where they are sent to the vocoder at the constant output rate. However, a frame at the head of the jitter buffer may be deleted by the jitter buffer manager, or a frame at the head of the jitter buffer may be delayed as an additional frame is inserted ahead of it.

The frame input process **240** and the frame output process **250** will be described in more detail below with reference to FIGS. **4** and **5** respectively. In brief, the frame input process receives frames from the IP socket **210** (along with the sequence number) and places the frames into the jitter buffer. Normally, the frame will be inserted in the next available slot in the buffer, however this depends on the sequence number of the received frame and the sequence number of frames in the jitter buffer. In an IP network, frames may be received out of order, in which case the frame input process will insert the frame into the appropriate slot in the jitter buffer in order to maintain its proper time alignment based on its sequence number. If the frame sequence number indicates that the frame is too late (because a frame with a later sequence number has already been sent to the vocoder) then the frame is discarded. The frame input process also detects if the jitter buffer depth has reached the high water mark, and if so, initiates a frame removal process. The frame removal process deletes one frame at a time until the desired jitter level is reached, as the high water mark represents a buffer condition indicating that the total delay added by the jitter buffer is getting too long and increasing delay beyond this point will lead to a degradation in quality of service. One advantage of the present invention is that the frame removal process does not randomly delete frames but rather looks for frames that satisfy a particular criteria, as indicated by a detector (in this example, the voice activity detector). This will be discussed in more detail below. The frame input process also triggers the frame output process when a new frame is received after the frame output process failed to service the last frame request issued by the vocoder (i.e. jitter buffer was empty at time of last request).

The frame input process uses the following inputs and outputs:

Inputs:	Type	Source
Voice Frame from IP Socket for Input to Jitter Buffer	Data	(IP Stack)
Frame Received (Sequence #)	Signal	(IP Stack)
"HighWaterMark"	Variable	(Predetermined)
"DesiredJitter"	Variable	(Predetermined)
"WaitingForFrame"	Boolean Variable	(Frame Output Process)
"NextFrame"	Variable	(Frame Output Process)
Outputs:	**Type**	**Destination**
"RemoveFrameCount"	Variable	(Frame Output Process)
"FrameAvailable"	Signal	(Frame Output Process)

The frame output process supplies frames to the vocoder (from the head of the jitter buffer) based on periodic requests by the vocoder. These periodic requests occur at the constant output rate determined by a local clock source. The frame output process also removes frames from the jitter buffer according to the frame removal process. The frame output process also detects a jitter buffer underflow and informs the input process accordingly. The frame output process also preferably detects if the jitter buffer depth has reached the low water mark and if so, initiates a silence frame insertion process by inserting frames at the head of the jitter buffer when the low water mark condition is true and when the voice activity detector indicates a silence frame has just been processed by the vocoder.

The frame output process uses the following inputs and outputs:

Inputs:	Type	Source
"RequestForFrame"	Signal	(Vocoder)
"SilenceDetected"	Signal	(VAD)
"RemoveFrameCount"	Variable	(Frame Input Process)
"FrameAvailable"	Signal	(Frame Input Process)
"LowWaterMark"	Variable	(Predetermined)
Outputs:	**Type**	**Destination**
"NextFrame"	Variable	(Frame Input Process)
"WaitingForFrame"	Boolean Variable	(Frame Input Process)
"BadFrameIndicator"	Signal	(Vocoder)
Voice Frame from Jitter Buffer to Vocoder	Data	(Vocoder)

FIG. **3** is a hardware block diagram illustrating the hardware components of a VoIP apparatus according to an embodiment of the invention for implementing the functional blocks of FIG. **2**. According to this embodiment, the hardware includes a microprocessor subsystem **300** and a Digital Signal Processor (DSP) subsystem **360**. The microprocessor subsystem and the DSP subsystem are interconnected via communication port **350**. The microprocessor subsystem **300** includes a microprocessor **320**, for example, an Advanced Risk Machine (ARM) processor **320**, RAM **330**, an address/data bus **325** and ROM **340**, as well as an Ethernet interface **310**. Note that the RAM represents working memory for implementing the jitter buffer and storing the values of variables whereas the ROM contains the real-time operating system (RTOS), the IP stack and the jitter buffer control software. Similarly, the DSP subsystem

includes a DSP **365**, RAM **370** and ROM **380** for containing software instructions for implementing, for example, the OS (the Operating System), the voice activity detector and the vocoder software. The DSP subsystem also includes an address/data bus **375**. The microprocessor **320** and the DSP **365** communicate via communication port **350**, which allows the transmission of both frames and signalling between the two subsystems. The DSP is also connected to a CODEC **160** for producing analog output to the receiver speaker on the receive side and also for receiving analog input from the microphone of the receiver. Note that this drawing only illustrates the components required to implement the functions of FIG. **2** and other components for implementing a fully-functional device will also be required, as should be apparent to a person skilled in the art. For example, the device can include a screen, keypad, and echo controller (which can include a switch loss system) for switching between receive mode, quiescent mode and transmit mode. Furthermore, the microprocessor subsystem **300** can act as a controller for the entire device or it can be an independent microprocessor subsystem controlled by the device controller.

In this embodiment, the microprocessor subsystem **300** implements the jitter buffer, the IP stack (accessed through the IP socket), the frame input process and the frame output process. The DSP subsystem **360** implements the vocoder and voice activity detector according to this embodiment of the invention. However, it should be apparent to a person skilled in the art that many different alternative implementations could be used, for example entire functionality could be implemented in one processor or individual pieces could be implemented in hardware (e.g. ASIC).

Both the frame input process **240** and the frame output process **250** represent control processes implemented by the microprocessor **320** for controlling the jitter buffer and for controlling the receipt and output of frames. Although the processes cannot be considered to be truly independent as each receives input from the other and sends output to the other, the two processes can be considered to run independently as the frame input process is governed primarily by the receipt of packets from the IP network and therefore depends on the packet reception rate. However, the frame output process is primarily governed by requests for frames from the vocoder (and thus indirectly from the CODEC) and therefore depends on the constant output rate (which is dictated by the clock frequency of the crystal **165** associated with the CODEC).

A block diagram illustrating the frame input process **240** is shown in FIG. **4**.

The frame input process runs at a variable rate dictated by the timing of received packets. Between frames, the process is effectively dormant and waits for the next frame **400** to be received from the IP network. The process wakes up when a frame received signal is received from the IP socket **410**. The process determines whether the received sequence number is less than the expected sequence number of the next frame. If so, this implies that the frame is too late for sending to the vocoder and is therefore discarded **430**, at which point the process returns to the wait-for-next-frame step **400**. Note the frame is also discarded if the received frame is already in the jitter buffer. Assuming the frame is not too late, the process determines whether the jitter buffer is full **440**. Note that the jitter buffer should not be full, especially if the two processes are operating correctly, and a full jitter buffer would indicate an unusual event has occurred, for example, a network error occurred resulting in a partial delay of several packets which are then all received simultaneously. However, a streaming service, for example, an internet radio broadcast, where there is no silence interval for a long time, can also fill the buffer. If the buffer is full, the frame is discarded. The processor then determines whether the jitter buffer is empty **450**. If the jitter buffer is empty, the processor determines whether the system is waiting for a frame **460** (in this embodiment, by evaluating a waiting-for-frame variable which is set by the frame output process). If the answer to this inquiry is negative, the processor adds the frame to the jitter buffer **470** and returns to the wait-for-next-frame step **400**. However, if the result of the waiting-for-frame inquiry is positive, the system not only adds the frame to the jitter buffer but sets the waiting for frame variable to false and also sends a frame available signal to the frame output process. The frame available signal causes the frame output process to expedite sending the frame to the vocoder. Note these signals need to be sent from the frame input process to the frame output process to trigger certain events, as the frame input process runs at a different rate than the frame output process.

Returning to step **450**, if the jitter buffer is not empty (which would be the normal state during a call), the processor adds a frame to the jitter buffer **480**. The processor then evaluates the condition of the buffer. In this example, the processor determines whether the jitter buffer is full beyond the high water mark level **485**. If the answer is negative, the processor simply returns to the wait-for-next-frame step **400**. However, if the jitter buffer is full beyond the high water mark level, the processor determines whether the RemoveFrameCount variable equals zero **490**, and if not, it returns to the wait-for-next-frame step. However, if the RemoveFrameCount is equal to zero, the frame input process sets the RemoveFrameCount variable to a number representing the number of frames the system will need to delete from the buffer in order to reach the desired jitter level. In this embodiment, this value is determined to be the high water mark minus the desired jitter level as shown at **495** and then returns to the wait-for-next-frame step.

FIG. **5** is a flowchart of the frame output process executed by the processor **320** according to an embodiment of the invention. In this embodiment, the process effectively lies dormant in a wait for signal state **500** until the processor receives a signal **505**. In this embodiment, the processor evaluates the signal to determine which of three possible signals is received. Thus the processor evaluated the received signal to determine whether a received signal is a request for frame signal from the DSP **510**; a frame available signal **530** from the frame input process; or a silence detected signal from the DSP **570**.

In response to receiving a request for frame signal, the processor determines whether the jitter buffer is empty **515**. If the jitter buffer is empty, the processor sets the waiting for frame variable to true **520** which is used to indicate to the frame input process that the DSP is waiting for a frame and returns to the wait for signal state. If the jitter buffer is not empty (which will be the normal state during a call) the processor determines the sequence number of the frame at the head of the jitter buffer **540**. The processor then determines whether this frame is late by evaluating whether the sequence number is equal to the value of the next frame variable **545**. The next frame variable indicates the expected sequence number of the next frame in the sequence of frames required by the vocoder. If the evaluation is negative (that is to; say the frame is late), the processor sends a bad frame indicator signal to the vocoder and increments the next frame variable **550** and then returns to the wait for signal state. If the sequence number is equal to the next

frame value, the processor gets the next frame from the jitter buffer and sends the frame to the vocoder, and increments the next frame variable **555**. The processor then checks the status of the jitter buffer to determine whether a condition that requires active management of the jitter buffer is true. In this embodiment, the process evaluates whether the jitter buffer is becoming depleted by determining whether the jitter buffer level is less than the low water mark value **558**. If not, the system simply returns to the wait for signal step. However, if the jitter buffer level is less than the low water mark, the system evaluates whether the insert frame count variable is equal to zero **560**. If it is, the processor sets the insert frame count variable to the number of frames which need to be inserted, which in this embodiment is equal to the desired jitter level minus the low water mark **565**. Thus, steps **558, 560** and **565** are evaluating a condition to indicate the status of the buffer and the number of frames that need to be inserted in order to achieve the desired jitter level. However, frames are only inserted where appropriate based on the detection of frames that satisfy particular criteria. This will be described in reference to steps **570, 580** and **590** below.

The evaluation step **530** determines whether the received signal was a frame available signal from the frame input process (which will happen if there is an outstanding frame request from the DSP which has not yet been satisfied). If so, the processor sets the waiting for frame variable equal to false **535** and then proceeds to step **540** as described above.

The frame output process is also responsible for the insertion and deletion of; frames when required and when frames satisfying the appropriate criteria are detected. Thus, the third possible signal received by the frame output process and evaluated at step **570** is a signal from the voice activity detector indicating that a silence frame is detected. If a silence frame is detected, the processor determines whether it should either insert a frame or delete a frame based on the condition of the buffer. This is implemented by evaluating whether the RemoveFrameCount variable is equal to zero (which would indicate that zero frames need to be removed) **575**. If the answer is negative, the processor gets the next frame from the jitter buffer and discards it on the assumption that the next frame will also be a silent frame. The processor also increments the next frame variable and decreases RemoveFrameCount variable **585** before returning to the wait-for-signal step.

If the RemoveFrameCount equals zero, the processor evaluates whether the insert frame count variable equals zero **580**. A positive evaluation at this step indicates that the system does not need to insert or delete a frame and thus returns to the wait-for-signal step **500**. However, if the insert frame count is not equal to zero, then the jitter buffer is becoming depleted and the system inserts an additional silence frame on the assumption that it is better to add a new silence after a silence frame than between two frames having actual speech content. Thus, the system proceeds to step **590** wherein: a silence frame is inserted at the head of the jitter buffer; the next frame variable is set to the sequence number of the inserted silence frame; and the processor decreases the insert frame count variable. The process then returns to the wait-for-signal step.

Thus, in this system, the jitter buffer manager, evaluates the condition of the buffer whenever a frame is received from the IP network or whenever a frame is requested by the vocoder. In this embodiment, there are two conditions, a high water mark which indicates that the jitter buffer is becoming full, and a low water mark which indicates that the jitter buffer is becoming depleted, as well as a desired jitter level. Indications are stored as to whether one of these conditions is satisfied. Meanwhile, the frame output process receives an indication from a detector whenever a frame satisfies a particular criteria. In this embodiment, the criteria is the receipt of a silence condition. If the jitter buffer level is greater than the high water mark and a silence condition is detected by the detector, the system discards the frame at the head of the jitter buffer. However, if the low water mark has been surpassed, the system inserts a silence frame after a silence frame is played out.

The values for the high water mark, desired jitter and low water mark may be hard-coded. Alternatively, these values may be dynamically updated by an optional Quality of Service (QoS) manager (not shown). For example, for systems which use a Real-time Transport Protocol (RTP) and a RTP control protocol (RTCP), the system preferably implements an adaptive jitter buffer that uses the RTCP protocol to keep track of RTP performance and updates these values dynamically in order to provide an optimal (minimal) jitter buffer delay while at the same time providing the desired QoS. In other words, the system monitors its performance (which may be dictated by the performance of the IP network which is transporting the packets) in order to adaptively alter the jitter buffer conditions in order to provide the best QoS as possible.

Note that the system as described operates on assumption that there will be a silence interval which is longer than a single frame. We will call this the silence assumption. Therefore, once high water mark is reached, the processor looks for a silence interval. Once a silence condition is detected, the processor deletes the next frame on the assumption that the next frame will also be silent. The silence assumption is reasonably accurate for two consecutive frames as there is a high probability that a silence interval will comprise more than one frame. However the probability that silence interval will continue diminishes with each successive frame. Thus the system preferably avoids deleting consecutive frames by waiting for the next silence interval to commence once it has deleted a frame (until RemoveFrameCount is decremented to zero). However, if the condition of the jitter buffer worsens, the system may have to delete more than one frame per silence interval.

FIG. **6** describes an alternative embodiment of the invention which implements a detector that evaluates frames before they are inserted into the jitter buffer in order to provide more control of the frames in the jitter buffer. Thus, a VoIP apparatus according to this embodiment would have a detector **640** coupled to the output of the IP interface and depacketizer **600**. This detector will detect frames which satisfy a criteria as they are received. This will allow the jitter buffer manager **650** to use more intelligence in controlling the insertion and/or deletion of frames in the jitter buffer. The apparatus also includes a jitter buffer **610**, a vocoder **660**, a digital-to-analog converter **670**, a sampling rate clock source **690** and a handset **680** (or alternatively, an interface circuit for coupling to an analog phone). In this embodiment, the detector **640** includes an additional vocoder **620** and an envelope detector **630** for detecting frames which represent silence as they are received. This will allow the jitter buffer manager **650** to more reliably know which frames in the buffer represent silence and can therefore insert or delete silence frames in a more flexible manner. For example, implementing the detector prior to playout from the buffer avoids having to operate under the above described silence assumption, as the system will know which frames are silent. Note that, as shown, two

vocoders are included (actually the detector only requires the decoder stage of a vocoder). However, given sufficient processing, a single multi-tasking vocoder can be used. Alternatively, the jitter buffer can store the output of the vocoder in systems for which conserving memory is less important than conserving DSP resources. Note that in such a system the decoded frames preferably maintain their sequence numbers while in the jitter buffer for proper handling of late frames, i.e. the decoded frames remain intact as a frame.

Yet another embodiment can use a different form of detector **640** for detecting frames which satisfy a different criteria. For example, many digital transmission systems encapsulate frames and/or packets with additional bits that are used for error detection and/or error correction. In such a system, the detector can determine which frames have bit errors (or uncorrectable errors). The jitter buffer manager can use this information in order to delete frames with errors if frames need to be deleted and no silence frames are available. For example, the IP stack, in particular the UDP protocol, which is used to transport the audio data, includes a checksum field that can be used to detect errors. Additionally, packet loss protection (PLP) algorithms similar to radio based error detection schemes have been proposed for IP networks. Such packet loss protection algorithms can provide suitable criteria for selecting frames with errors to be deleted.

Note that data transmission systems which use error detection and/or error correction typically use these bits to evaluate when a frame with errors is received in order to request retransmission of the errored frame. However, for voice conversations or other time-sensitive streaming services, there is typically insufficient time to request the retransmission of a frame. Typically, these frames are still sent to the vocoder on the assumption that there is insufficient time to replace the frame and that there is sufficient information in the frame to warrant the playback of the frame rather than discarding the frame. However, if the high water mark has been reached such that the system will need to discard a frame, the jitter buffer manager preferably selects a frame which is received with errors to be discarded, rather than discarding a frame at random.

Numerous modifications, variations and adaptations may be made to the particular embodiments of the invention described above without departing from the scope of the invention, which is defined in the claims. For example, we have described a system having a single microprocessor for implementing both the-frame input process and the frame output process. As an alternative, each process can be implemented by a dedicated processor, or the DSP and Microprocessor functions can be implemented using a single processor, or some of the software blocks can be implemented in hardware blocks. Furthermore the digital envelope detector could be replaced with an analog detector (e.g. RC circuit) located after the CODEC.

Additionally, the criteria to delete a frame can include mode information,(e.g. transmit, receive or quiescent mode), associated with received frames. In an embodiment using such mode information, the voice activity detector is used by an echo controller (not shown) to determines when said apparatus is operating in transmit or quiescent mode and the criteria includes information associated with received frames about the mode of operation of the apparatus. Alternatively, the echo controller can use a separate detector and send mode information to the Buffer Manager.

Furthermore, we have described a system that adds or deletes entire frames. However as an alternative, the system could be configured to add or delete partial frames. For example, when the low water mark is reached, the system can add a period of silence shorter than a frame instead of adding a complete silence frame.

What is claimed is:

1. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria; and

a buffer manager for controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria;

wherein said detector is a silence detector and criteria represents a silence interval, and wherein said silence detector comprises an envelope detector.

2. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria; and

a buffer manager for controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria;

wherein said detector is a silence detector and criteria represents a silence interval and wherein said silence detector comprises a voice activity detector.

3. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria; and

a buffer manager for controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria;

wherein said apparatus includes an echo controller and wherein said detector determines the mode of operation of said apparatus and wherein said criteria includes information associated with received frames about the mode of operation of the apparatus.

4. Apparatus comprising:

data interface for receiving frames from a data network;

jitter buffer memory for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either the insertion or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending upon said buffer condition;

wherein said detector is a silence detector and said criteria represents a silence interval.

5. Apparatus as claimed in claim 4 wherein said silence detector comprises an envelope detector.

6. Apparatus as claimed in claim 4 wherein said silence detector comprises a voice activity detector.

7. Apparatus comprising:

data interface for receiving frames from a data network;

jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either the insertion or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said apparatus includes an echo controller and wherein said detector determines the mode of operation of said apparatus and wherein said criteria includes information associated with received frames about the mode of operation of the system.

8. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either of the insertion of or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said detector comprises a bit error detector and said criteria comprises a frame received with errors.

9. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either of the insertion of or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said buffer manager includes separate Frame Input and Frame Output process which operate at different rates.

10. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either of the insertion of or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said detector detects said criteria after playout and said buffer manager deletes or inserts a single frame per silence interval at the head of said jitter buffer.

11. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either of the insertion of or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said detector detects said criteria prior to playout and said buffer manager deletes or inserts frames based on frame content.

12. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either of the insertion of or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said detector detects said criteria prior to playout and said buffer manager deletes frames which satisfy the condition prior to playout.

13. Apparatus comprising:

a data interface for receiving frames from a data network;

a jitter buffer for temporarily storing said frames;

a detector for detecting frames which satisfy a criteria indicative of the impact of the insertion or deletion of a frame; and

a buffer manager for controlling the input of received frames into said jitter buffer and the output of frames from said jitter buffer, said buffer manager capable of determining a buffer condition which requires either of the insertion of or deletion of a frame from said jitter buffer, and inserting or deleting frames in said buffer which satisfy said criteria depending on said buffer condition;

wherein said condition comprises a high water mark and a low water mark and wherein said buffer manager deletes a frame from the head of said jitter buffer when the high water mark is exceeded and when the criteria is satisfied and inserts a frame at the head of said jitter buffer when the buffer is depleted below the low water mark and when said criteria is satisfied.

14. A method of managing a jitter buffer comprising the steps of:

receiving frames from a data network;

storing received frames into said jitter buffer;

detecting frames which satisfy a criteria; and

controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria;

wherein said condition comprises a high water mark and a low water mark and wherein said controlling step comprises:

deleting a frame from said jitter buffer when the high water mark is exceeded and when said criteria is satisfied; and

inserting a frame into said jitter buffer when said buffer is depeleted below said low water mark and said criteria is satisfied.

15. Apparatus comprising:

jitter buffer;

data interface for receiving frames from a data network and inserting said frames into said jitter buffer;

a detector for indicating a frame which represents a silence interval;

a buffer manager for determining a buffer condition, and, responsive to said condition being satisfied, controlling the insertion and deletion of frames in said buffer responsive to the detection of a silence interval.

16. Apparatus as claimed in claim **15** wherein said data interface comprises a depacketizer for deconstructing frames from a packet based data network.

17. Apparatus as claimed in claim **15** wherein said buffer manager inserts received frames into buffer in the correct order according to their sequence number.

18. Apparatus as claimed in claim **15** wherein said buffer manager outputs frames at a constant rate determined by a local clock source which can be in digital phone or in integrated phone or in interface to an analog/digital phone.

19. A method of managing a jitter buffer comprising the steps of:

receiving frames from a data network;

storing received frames into said jitter buffer;

detecting frames which satisfy a criteria using envelope detection means; and

controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria;

wherein said criteria represents a silence interval.

20. A method of managing a jitter buffer comprising the steps of:

receiving frames from a data network;

storing received frames into said jitter buffer;

detecting frames which satisfy a criteria including detecting voice activity; and

controlling the frames stored in said jitter buffer based on the condition of said buffer and on frames which satisfy said criteria;

wherein said criteria represents a silence interval.

* * * * *

BMace_Job_1_of_1

Printed by HPS Server

for

EAST

Printer: cpk2_3b05_gbgjptr

Date: 05/25/04

Time: 08:53:18

Document Listing

Document	Selected Pages	Page Range	Copies
US20040038691	15	1 - 15	1
Total (1)	15	-	-

US005526353A

United States Patent [19]

Henley et al.

[11] Patent Number: 5,526,353

[45] Date of Patent: Jun. 11, 1996

[54] **SYSTEM AND METHOD FOR COMMUNICATION OF AUDIO DATA OVER A PACKET-BASED NETWORK**

[76] Inventors: **Arthur Henley**, 10705 Bay Laurel Trail, Austin, Tex. 78750; **Scott Grau**, 13303 Ivywood Cove, Austin, Tex. 78729

[21] Appl. No.: **359,393**

[22] Filed: **Dec. 20, 1994**

[51] **Int. Cl.**[6] **H04L 12/64**

[52] **U.S. Cl.** **370/60.1**; 370/79; 370/94.2; 370/81; 370/100.1; 370/118; 375/241; 375/371

[58] **Field of Search** 370/60, 60.1, 61, 370/79, 94.1, 94.2, 100.1, 81, 105.3, 118; 340/825.2, 825.21; 379/93; 375/240, 241, 371

[56] **References Cited**

U.S. PATENT DOCUMENTS

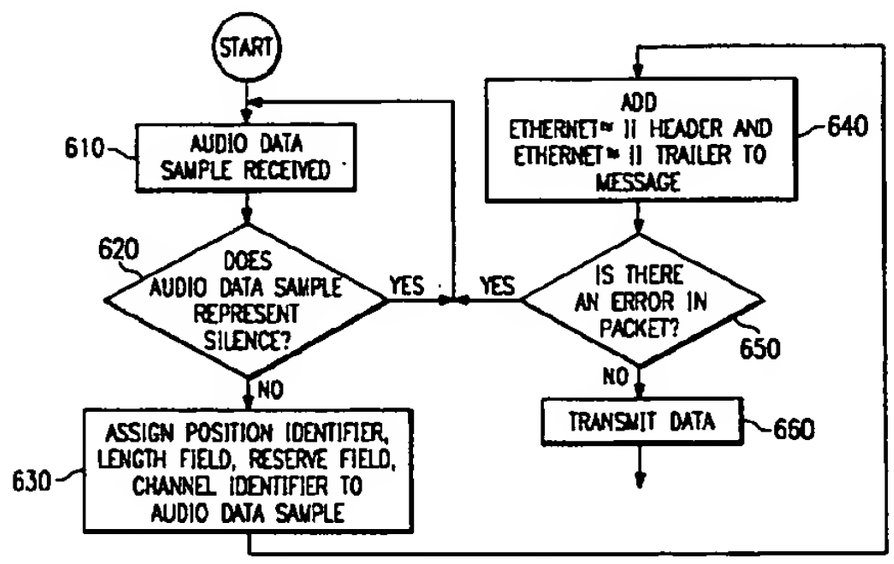
4,507,782	3/1985	Kunimasa et al.	371/32
4,712,214	12/1987	Meltzer et al.	371/32
4,841,526	6/1989	Wilson et al.	371/32
4,870,661	9/1989	Yamada et al.	375/240
4,885,749	12/1989	Golden	371/32
4,888,767	12/1989	Furuya et al.	370/95.2
4,908,828	3/1990	Tikalsky	371/69.1
4,947,484	8/1990	Twitty et al.	371/37.1
4,970,714	11/1990	Chen et al.	370/17
5,010,553	4/1991	Scheller et al.	371/35
5,084,877	1/1992	Netravali et al.	371/32
5,103,467	4/1992	Bedlek et al.	375/371
5,148,429	9/1992	Kudo et al.	370/94.2
5,168,497	12/1992	Ozaki et al.	370/94.1
5,191,583	3/1993	Pearson et al.	370/94.1
5,287,182	2/1994	Haskel et al.	375/376 X
5,412,642	5/1995	Nunokawa	370/17

Primary Examiner—Douglas W. Olms
Assistant Examiner—Russell W. Blum
Attorney, Agent, or Firm—Hitt Chwang & Gaines

[57] **ABSTRACT**

A system and method for communicating audio data in a packet-based computer network wherein transmission of data packets through the computer network requires variable periods of transmission time. The system comprises: (1) a packet assembly circuit for constructing a data packet from a portion of a stream of digital audio data corresponding to an audio signal, the packet assembly circuit generating a position identifier indicating a temporal position of the portion relative to the stream, inserting the position identifier into the data packet and queuing the data packet for transmission through a backbone of the computer network and (2) a packet disassembly circuit, having a buffer associated therewith, for receiving the data packet from the backbone, the packet disassembly circuit inserting the portion into an absolute location of the buffer, the position identifier determining the location, the portion thereby synchronized with adjacent portions of the stream of digital audio data in the buffer to compensate for the variable periods of transmission time.

40 Claims, 3 Drawing Sheets



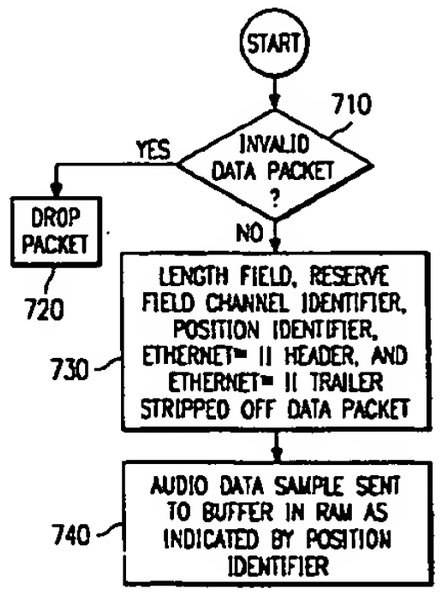
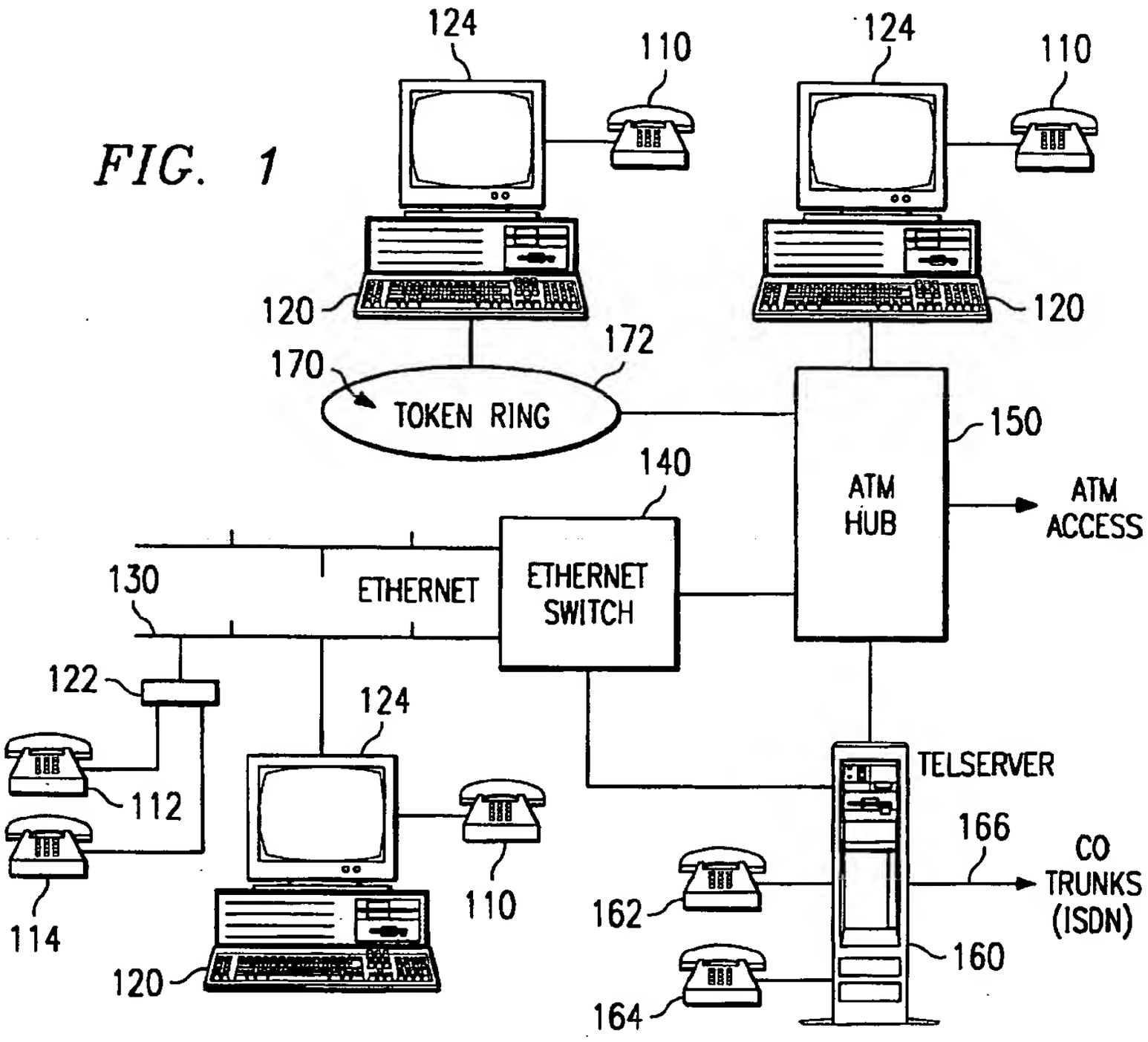


FIG. 1


124
110
124
110
120
120
170
172
TOKEN RING
150
ATM HUB
ATM ACCESS
140
ETHERNET SWITCH
130
ETHERNET
122
124
112
114
120
110
TELSERVER
166
CO TRUNKS (ISDN)
162
160
164

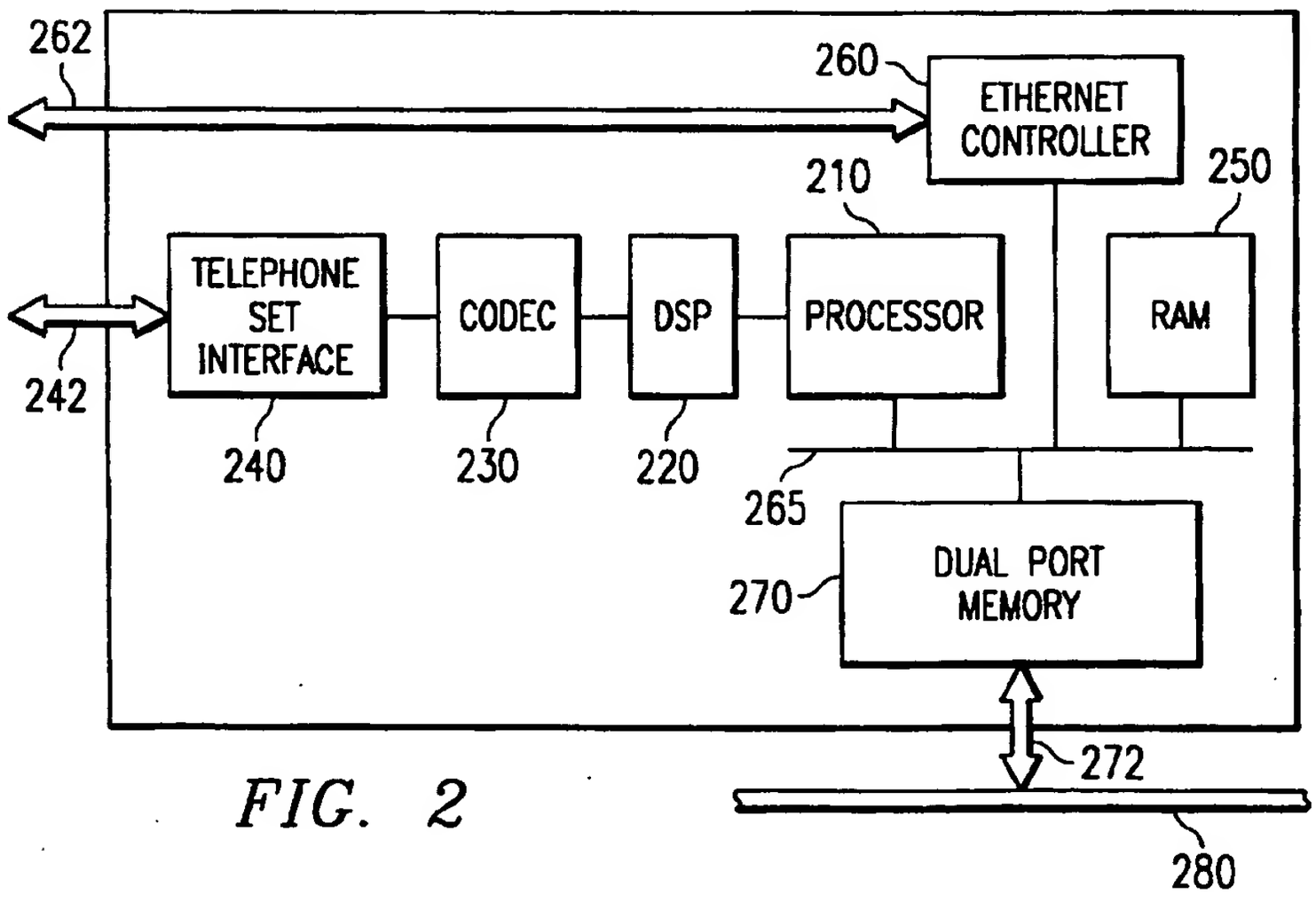

262
260
ETHERNET CONTROLLER
210
250
TELEPHONE SET INTERFACE
CODEC
DSP
PROCESSOR
RAM
242
240
230
220
265
270
DUAL PORT MEMORY
272
280

FIG. 2

FIG. 3

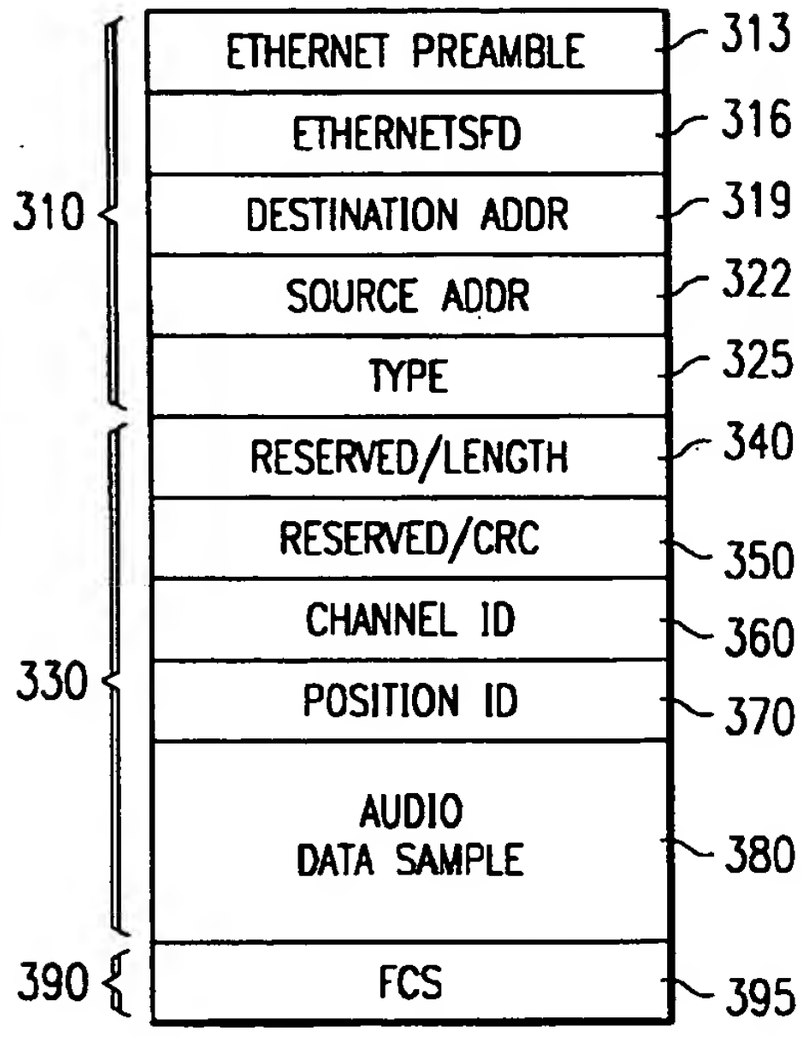
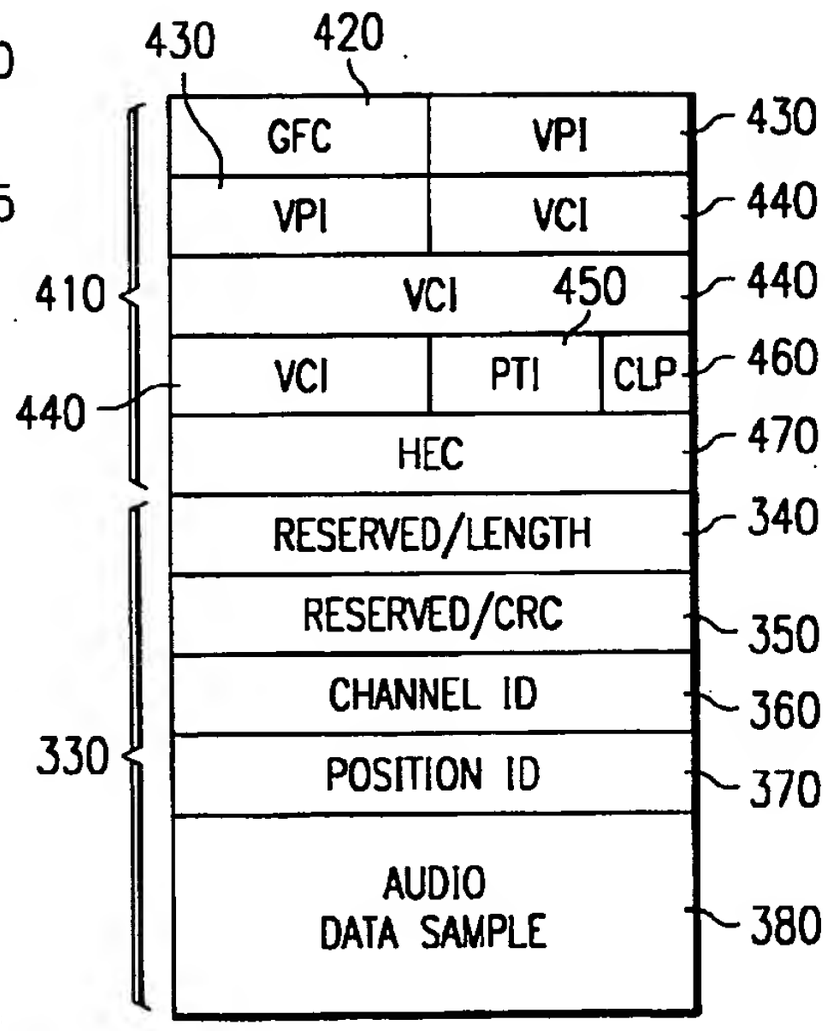


FIG. 4



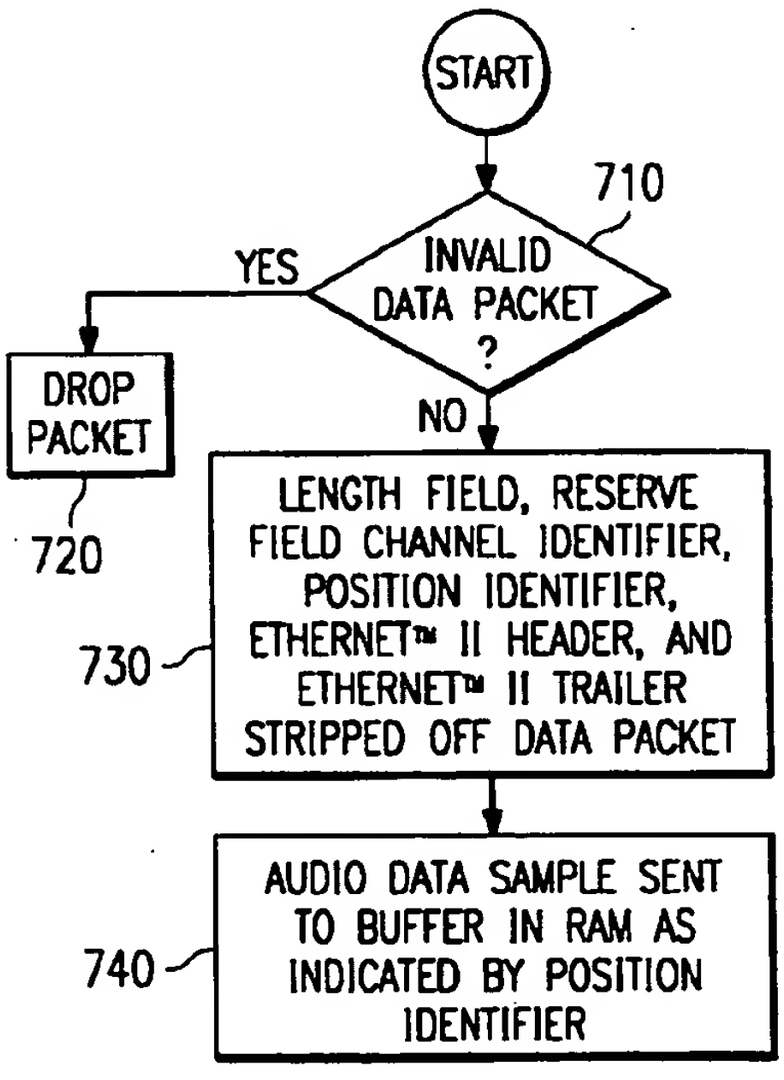


FIG. 7

FIG. 5

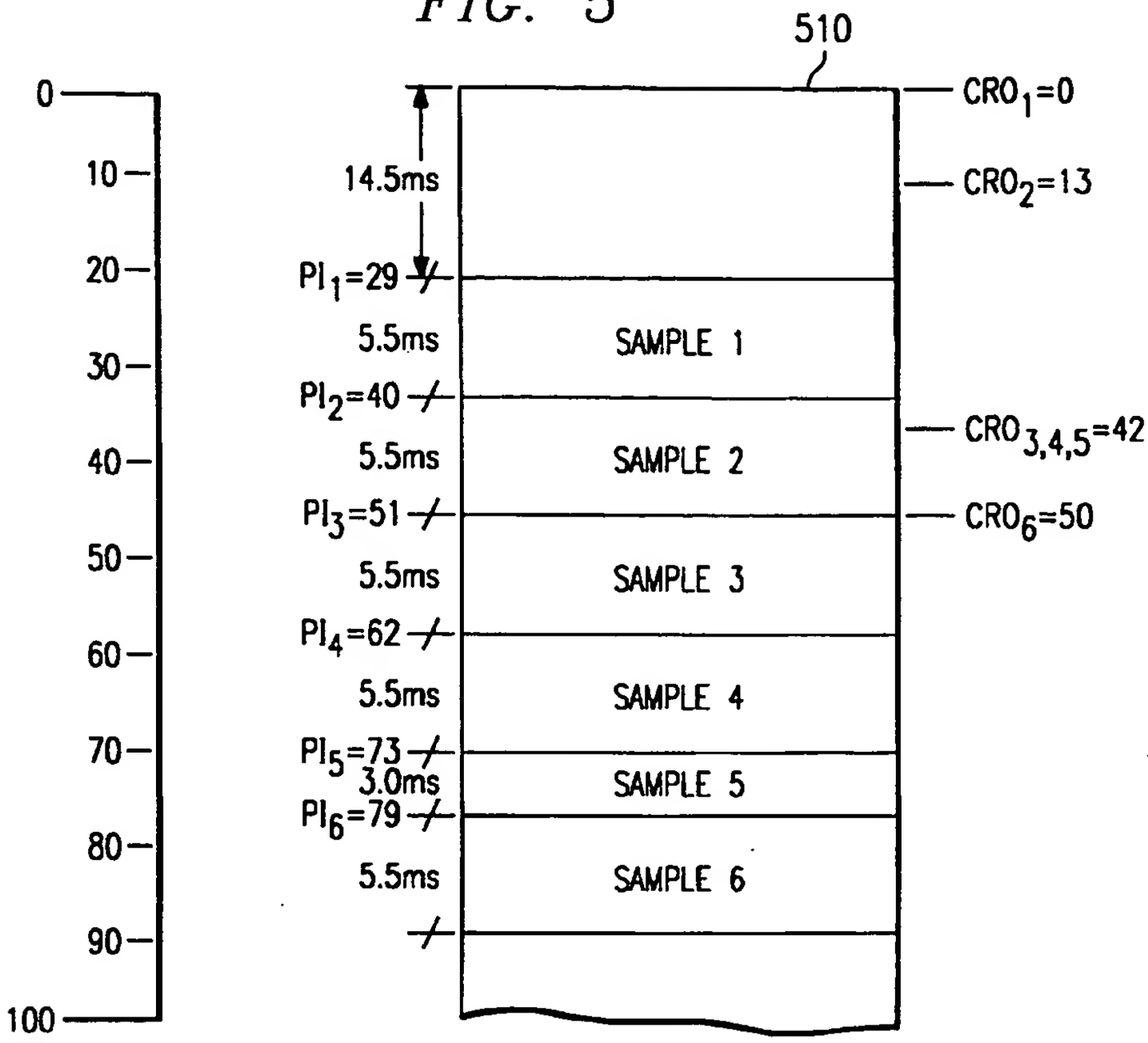
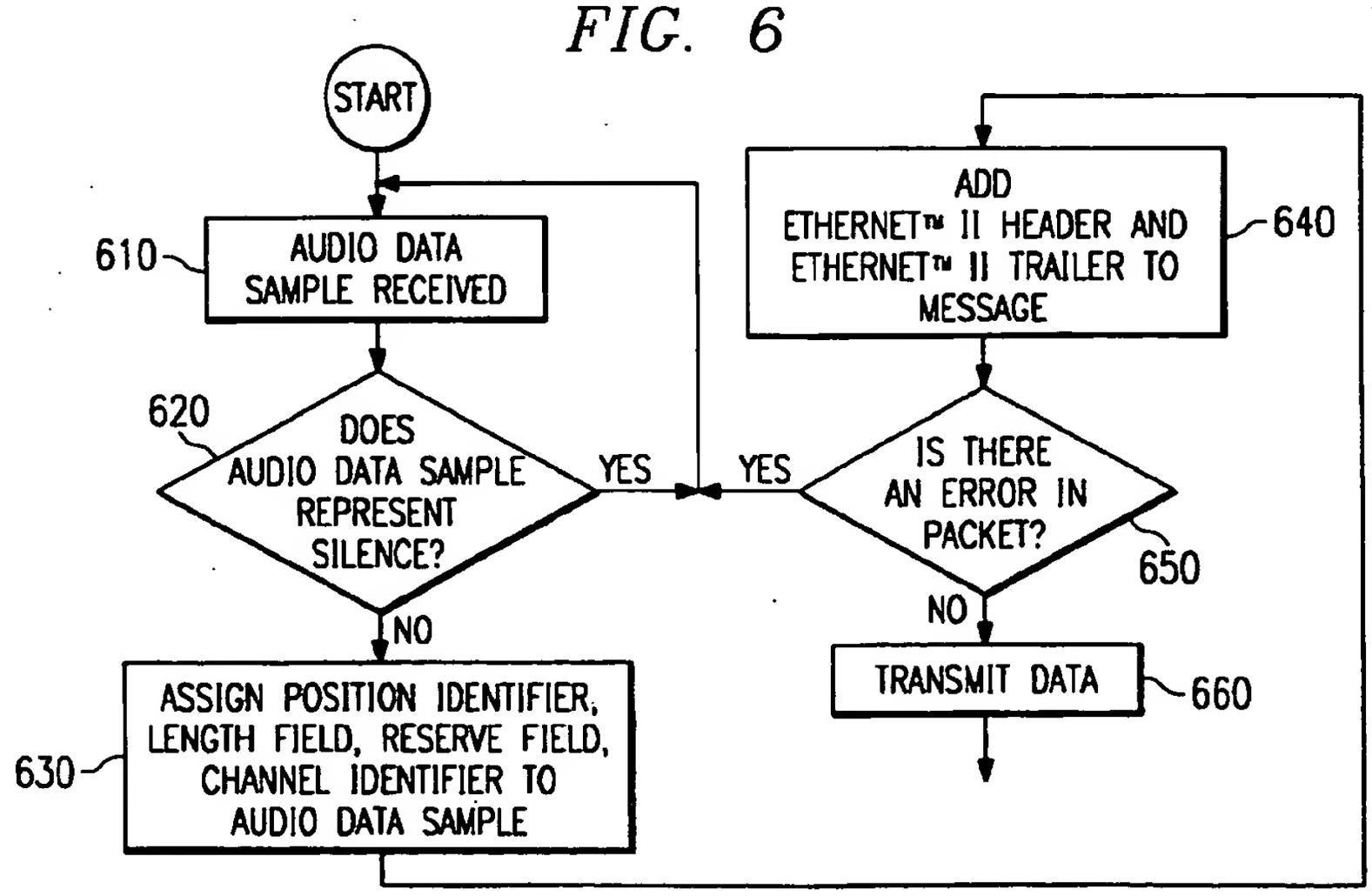

510
0
10
20
30
40
50
60
70
80
90
100
14.5ms
PI1=29
5.5ms
PI2=40
5.5ms
PI3=51
5.5ms
PI4=62
5.5ms
PI5=73
3.0ms
PI6=79
5.5ms
SAMPLE 1
SAMPLE 2
SAMPLE 3
SAMPLE 4
SAMPLE 5
SAMPLE 6
CRO1=0
CRO2=13
CRO3,4,5=42
CRO6=50

FIG. 6


START
610
AUDIO DATA SAMPLE RECEIVED
620
DOES AUDIO DATA SAMPLE REPRESENT SILENCE?
YES
NO
630
ASSIGN POSITION IDENTIFIER, LENGTH FIELD, RESERVE FIELD, CHANNEL IDENTIFIER TO AUDIO DATA SAMPLE
640
ADD ETHERNET™ II HEADER AND ETHERNET™ II TRAILER TO MESSAGE
650
IS THERE AN ERROR IN PACKET?
YES
NO
660
TRANSMIT DATA

SYSTEM AND METHOD FOR COMMUNICATION OF AUDIO DATA OVER A PACKET-BASED NETWORK

TECHNICAL FIELD OF THE INVENTION

The present invention is directed, in general, to computer networks and, more specifically, to a system and method for transmitting and receiving digitized audio data in a packet-based computer network to compensate for variable packet transmission times (jitter).

BACKGROUND OF THE INVENTION

Historically, entirely separate communication systems have been employed to transmit audio data (sometimes referred to shorthandedly as "voice" for simplicity's sake) and computer data (sometimes abbreviated "data" for a like purpose, although it should be understood that "voice" data and computer "data" both fall within the broad definition of "data").

Over a century ago, analog telephone networks were developed to carry analog audio signals. Telephone networks allow communication of audio data, or more broadly, audio signals between two or more users by establishing, with central switching equipment, a dedicated communication circuit or "channel" among the users. Because the channel, once established, is dedicated exclusively to transmission of the users' conversation, the conversation is not required to compete for the channel's bandwidth with other conversations. The advantage of having a dedicated channel per conversation is that any transmission delays from speaker to listener is purely a function of the unfettered speed of the audio signal through the telephone network. Since this speed does not significantly vary over time, such dedicated channels are capable of providing "isochronous" transmission. Unfortunately, one significant disadvantage of dedicated channels is that they require significant bandwidth; that is, the complete bandwidth of the channel remains available and dedicated to carriage of the conversation even when no audio information is being transmitted.

In recent years, efforts have been underway to establish interface standards for digital transmission of audio signals over telephone networks. The most noted of the existing standards is the Integrated Services Digital Network ("ISDN") significantly sponsored by AT&T. ISDN standardizes connection interfaces, transmission protocols and services to create a unified digital circuit switching network. More recently, recommendations for Broadband ISDN ("BISDN") have been adopted. Unlike ISDN, which is a digital network standard, BISDN uses packet relay, or Asynchronous Transfer Mode ("ATM") as a transmission standard, and is of particular importance in transmission over broadband "backbones" and, in particular, fiber optic lines. ATM is primarily a connection-oriented technique that can transport both connection and connectionless-oriented services at either a constant bit rate or a variable bit rate. ATM provides bandwidth on demand and handles all traffic types through fast-packet switching techniques that reduce the processing of protocols and uses statistical multiplexing.

In ATM, audio data are split into relatively small blocks or packets, commonly called "cells." The cells are individually communicated through the ATM network by transmitters and receivers that are not synchronized. Networks limited to synchronous transmission generally require dedicated channels and a clock to control the synchronous transmission of audio data through the network. Therefore,

ATM allows telephone networks to depart from the above-described synchronous transmission of audio data over dedicated, isochronous channels, thereby dramatically increasing network efficiency by combining previously dedicated channels and decreasing cost by eliminating synchronicity. Both ISDN and BISDN therefore hold much promise for the future. However, widespread application of these standards has been slow, as the installed base of analog equipment (including telephone sets) is substantial and presents great resistance to change.

Packet transmission or ATM should not be confused with TDM. TDM calls for synchronous division of the overall bandwidth of a common backbone into multiple low speed channels and assigns a specific time slot to each channel. In other words, if there are four channels, each channel is allocated a fourth of the bandwidth. The bandwidth is systematically switched, such that channel 1 gets its fourth-bandwidth, followed by channels 2, 3, 4, 1, 2 and so on. In TDM, the processing power necessary to share common bandwidth is located in various, centralized multiplexers. This centralization is acceptable if channel traffic is constant or predictable. However, when traffic occurs in short intervals (as in the real world), processing becomes nontrivial, resulting in an effective loss of bandwidth.

In contrast, packet transmission or ATM is asynchronous, allocating the total backbone bandwidth on an as-demanded basis. For instance, if channel 1 is highly active, it may receive more than its pro-rata share of overall bandwidth. When channel 1's activity declines, its allocated bandwidth likewise declines. Thus, packet transmission or ATM is most adept at handling "bursty" transmission of data, wherein the activity of each individual channel is subject to relatively wide variation. Thus, because computers transmit data through networks in packets, computer data are said to be "bursty." Unlike TDM, the processing power required to create, transmit and receive packets is distributed among all of the communicating devices, rather than being centralized. Thus, bandwidth is not effectively lost due to inherent limitations in centralized processing.

Although telephone networks have been in place for over a century, computer networks have come into being only in the past quarter century. In contrast with the dedicated channels of traditional telephone networks, computer networks allow individual computers shared access to a common communication backbone having relatively broad bandwidth (in a manner quite similar to ATM).

As in ATM, computer data are divided into packets, each of which includes error protection. The individual networked computers ("nodes") thus are granted access to the complete bandwidth of the backbone so they can transmit their packets of computer data thereon. When the transmitting computer completes transmission of the packet, the backbone is made immediately available for the other computers.

A special case of a computer network is a personal computer ("PC") network. Whereas PCs were once only used as isolated devices, they are now used for a wide range of applications requiring the PCs to communicate with each other over a computer network.

Today, networking in a large office with hundreds of PCs, or in a small office with just a few PCs, is very popular and, quite simply, the best way to share data and communicate among PCs. A local area network ("LAN") is a specific type of network connecting PCs located in relatively close proximity. A wide area network ("WAN") is a network of separate LANs. The backbones of such LANs typically comprise coaxial or twisted-pair cable.

All networks experience delay in end-to-end data transmissions therethrough. This delay (termed "latency") affects the overall efficiency and effective bandwidth of the network. ATM and computer networks, because they are asynchronous, are further subject to "jitter," defined as change in network latency as a function of time. Jitter is largely unpredictable; however, the overall quantity of traffic on a network tends to increase both latency and jitter.

At the heart of any computer network is a communication protocol. A protocol is a set of conventions or rules that govern the transfer of data between computer devices. The simplest protocols define only a hardware configuration, while more complex protocols define timing, data formats, error detection and correction techniques and software structures.

Computer networks almost universally employ multiple layers of protocols. A low-level physical layer protocol assures the transmission and reception of a data stream between two devices. Data packets are constructed in a data link layer. Over the physical layer, a network and transport layer protocol governs transmission of data through the network, thereby ensuring end-to end reliable data delivery.

The most common physical networking protocol or topology for small networks is Ethernet, developed by Xerox. When a node possesses a packet to be transmitted through the network, the node monitors the backbone and transmits when the backbone becomes clear. There is no central backbone master device to grant requests to gain access to the backbone. While this type of multipoint topology facilitates rapid transmission of data when the backbone is lightly utilized, packet collisions may occur when the backbone is heavily utilized. In such circumstances, there is a greater chance that multiple nodes will detect that the backbone is clear and transmit their packets coincidentally. If packets are impaired in a collision, the packets are retransmitted until transmission is successful.

Another conventional physical protocol or topology is Token Ring, developed by IBM. This topology employs a "token" that is passed unidirectionally from node to node around an annular backbone. The node possessing the token is granted exclusive access to the backbone for a single packet transfer. While this topology reduces data collisions, the latency incurred while each node waits for the token translates into a slower data transmission rate than Ethernet when the network is lightly utilized.

Several network and transport protocols designed to handle bursty data transmission are well known in the art. One protocol that enables communication between PCs is the Microcom Networking Protocol ("MNP"), developed by Microcom Systems. MNP is suited for both interactive communication and file transfers and may be implemented on a wide range of computers. MNP packets data with a header and trailer containing packet type, CRC and other information concerning the packet. While the MNP protocol provides relatively error-free transmission of data, the significant overhead of the header and trailer decreases data bandwidth.

The prior art includes many techniques involving manipulation of data to boost the data transmission rate or "throughput" of a network. U.S. Pat. No. 4,691,314, assigned to Microcom, discloses a system for transmitting data in larger, adjustable-sized packets. Because the system allows for larger packets, relatively less header and trailer overhead is required.

However, when the transmission medium is unreliable (such as when the data are transmitted over noisy telephone network lines), errors may occur more frequently in the data. As packet length increases, the chance of corruption of the data within the packet also increases. Furthermore, the larger packets must be retransmitted, thereby decreasing network throughput.

Another network and transport protocol is Transmission Control Protocol/Internet Protocol ("TCP/IP"). This protocol employs a "go back N method" of error and flow control over a datagram network. In a "go back N method" of error control, if there is a transmission error, a packet loss, excessive latency in the delivery of a packet, delivery of a packet out of sequence or an overflow of a receiver buffer, significant loss of throughput is realized due to excessive packet retransmissions.

As the domain of digital computer networks continues to expand, the networks are challenged with new and more difficult responsibilities. One of those challenges is multimedia. In recent years, there have been a number of attempts to produce a digital data network additionally capable of carrying data representing a digitized audio signal (again, "voice"), thereby additionally functioning as a telephone network and, in sum, yielding a so-called "multimedia network."

As described above, however, audio signals are extremely time-sensitive, because users are extremely sensitive to minute tones, inflections and pauses, particularly in human speech. Thus, a computer data network that also must transmit audio data is forced to cope with the communication of both bursty computer and time-sensitive audio data on the backbone.

The repercussion is that the above-described data network and transport protocols that are sufficient to transmit data are insufficient for transmission of time-sensitive audio data. The latencies present in a communication network, e.g., those relating to coding, packet assembly, media access, propagation, receiver buffering and decoding, must be precisely compensated for to preserve the fidelity of the audio signal.

At this point, an interesting observation should be made. Data has been described above as being bursty. It has been implied that audio data is somehow not. Both of these assumptions prove to be inaccurate. First, data is only bursty because computer networks have been dealing with it in that manner for so many years. In fact, once transmission of a batch of data begins, data transmission rate is constant. Second, because spoken words are made of small, discrete utterances (syllables or words), audio data is inherently bursty. Therefore, while it is certainly true that audio data is extremely time-sensitive, audio data is likewise bursty. If a way can be found to compensate for network jitter, audio data should be highly amenable to packet-based transmission.

Therefore, what is needed in the art is a system and method for transmitting and receiving digitized audio data in a packet-based network to adjust for variable packet transmission times. The system and method must deliver end-to-end reliable transmission of data, accounting for all delays in the transmission network while presenting high fidelity audio signals at the receiving end.

SUMMARY OF THE INVENTION

To address the above-discussed deficiencies of the prior art, it is a primary object of the present invention to compensate for jitter in a computer network to provide high fidelity transmission of audio data through the network.

In the attainment of the above primary object, the present invention provides a system and method for communicating audio data in a packet-based computer network wherein transmission of data packets through the computer network requires variable periods of transmission time. The system comprises a packet assembly circuit for constructing a data packet from a portion of a stream of digital audio data corresponding to an audio signal. The packet assembly circuit generates a position identifier indicating a temporal position of the portion relative to the stream, inserting the position identifier into the data packet and queuing the data packet for transmission through a backbone of the computer network.

The system further comprises a packet disassembly circuit, having a buffer associated therewith, for receiving the data packet from the backbone. The packet disassembly circuit inserts the portion into an absolute location of the buffer, the position identifier determining the location, the portion thereby synchronized with adjacent portions of the stream of digital audio data in the buffer to compensate for the variable periods of transmission time.

Transmission of audio data over a computer network is a more exacting task than transmission of less time-sensitive computer data. As previously described, audio data are extremely time sensitive; and as a result, the system hardware, software and transport protocol must be precisely coordinated to realign the audio data at the receiving end. The present invention provides such a system and method for ensuring high fidelity and clear transmission of audio data through a computer network.

The position identifier of the present invention should not be confused with a packet sequence number. As will be described in more detail, the position identifier points to a specific, absolute address in the buffer and not to a position of the packet relative to other packets. With sequence numbers, one may only discern that packet **3** follows packet **2** and precedes packet **4**. With the position identifier, one may further discern vital packet synchronization information: that packet **3** follows packet **2** by, e.g., 5 milliseconds ("ms") and precedes packet **4** by, e.g., 15 ms. In distinct contrast to sequence numbers, position identifiers may cause portions of packets to occlude (and therefore overwrite) portions of other packets, may result in temporal gaps between packets (resulting in interstitial periods of silence) and allow packets to be transmitted in an arbitrary order without compromising relative packet synchronization.

In a preferred embodiment of the present invention, the system further comprises an interpolation circuit for inserting synthesized audio data into a designated location of the buffer to thereby lengthen the portions of the stream of audio data in the buffer. The interpolation circuit addresses those circumstances in which the length of the buffer decreases during reception of audio data from the backbone. This happens when data are read from the buffer faster than they are written to the buffer.

For example, if the clock of a coder/decoder ("CODEC") that reads from the buffer is too fast, the CODEC reads too rapidly and the buffer becomes too short. The interpolation circuit is adapted to detect when the buffer is too short and adjust the buffer toward a predetermined length by adding the synthesized audio data. The interpolation circuit ensures that buffer stays close to its predetermined length for efficient realignment of the audio data in the buffer.

The system of the present invention further comprises a decimation circuit for deleting audio data from a designated location of the buffer to thereby shorten the portions of the stream of audio data in the buffer. The decimation circuit addresses the circumstance in which the length of the buffer increases during reception of audio data from the backbone. This happens when data are read from the buffer slower than they are written to the buffer.

For example, if the CODEC clock triggers too slowly, or if the audio data are transmitted at an excessive rate through the LAN, the buffer window lengthens. The decimation circuit is adapted to detect when the buffer is too long and adjust the buffer toward its predetermined length by deleting selected audio data. Like the interpolation circuit, the decimation circuit ensures that buffer stays close to its predetermined length for efficient realignment of the audio data in the buffer.

In a preferred embodiment of the present invention, the data packet of the present invention comprises source and destination fields for determining a transmission route of the data packet through the computer network. This embodiment is primarily directed to an Ethernet environment, wherein each node in the computer network is designated by a specific address. Prior to routing the audio data across the backbone of the computer network, the data packet is assigned a source and destination address designating the appropriate nodes. Alternatively, a channel identifier may be used in WAN applications (via ATM) to ensure accurate delivery.

As previously described, packet-based transmission allows advantageous distributed call processing and signalling. Thus, each packet assembly circuit is individually responsible for determining the routing of the audio data through the network.

In a preferred embodiment of the present invention, a value of the position identifier is a function of a length of a portion of the stream of digital audio data in a previously-transmitted data packet. Thus, the position identifier preferably designates the position at which the first datum of each portion is to be placed in the buffer. That position preferably follows the position of the last datum of the previously-transmitted data packet.

In a preferred embodiment of the present invention, each portion of audio data (a "sample") is placed in a data packet having a prescribed length. In addition to the sample, the data packet contains a position identifier. The position identifier directs the samples into absolute positions in the buffer, that may or may not be successive. The distinct advantage of the position identifier is temporal synchronization of samples in the buffer.

It should also be understood that other than audio data can occupy the data packet. Given a special header designation, signalling and call processing (control) data can be loaded into a packet. Again, this allows for distributed, decentralized processing. Once loaded into a packet, the control data is treated no differently than audio data in its travels through the network.

In a preferred embodiment of the present invention, a length of a travelling window within the buffer of the present invention is about 20 ms. The window is defined as the difference between the locations at which data are written to and read from the buffer. The window is established at that optimal length (in an Ethernet application) as a function of packet length and network characteristics (such as latency in packet assembly, media access, transmission and disassembly). In an ATM network, window length should also be about 20 ms. With the Internet, window length should be about 50–100 ms to account for significant latency in that very large network. In each case, if the window were to be

shorter, there may not be sufficient time to allow for the latency. Echo cancellation is typically a requirement when the round trip audio delay exceeds 60 ms.

In a preferred embodiment of the present invention, the data packet is capable of containing a portion having a length of about 5.5 ms. The length of about 5.5 ms corresponds to a 44 byte pulse code modulated ("PCM") audio data sample. Again the 5.5 ms length is adjustable and depends upon network characteristics. Also, the length of the portion is as-compressed. Since many compression algorithms are variable, the uncompressed length may vary.

In a preferred embodiment of the present invention, the system further comprises a digital conversion/compression circuit, coupled to the packet assembly circuit, for digitizing and compressing the audio signal into the stream of digital audio data. Again, many compression algorithms are variable, so there is not a linear correspondence between uncompressed and compressed data length.

The digital conversion/compression circuit converts the analog audio signal into a stream of digital audio data for use by the packet assembly circuit. The packet assembly circuit arranges the audio data into data packets for transmission across the backbone. The advantage of digitizing and compressing the data is that larger effective bandwidth is thereby available for transporting audio data through the computer network.

In a preferred embodiment of the present invention, the system further comprises a decompression/analog conversion circuit, coupled to the packet disassembly circuit, for decompressing and converting the stream of digital audio data back into the audio signal. Thus, the received audio data are converted into a medium that the listener on the receiving end can understand and respond to in kind.

In a preferred embodiment of the present invention, the computer network of the present invention comprises a plurality of computers coupled to the backbone, the packet assembly circuit and the packet disassembly circuit located in separate ones of the computers. Thus, present invention is designed to operate in a computer network having a plurality of nodes and able to support many ongoing telephone conversations. The computer network may be of a client-server or peer-peer topology. Thus, the system of the present invention allows a computer network to supplant a private branch exchange ("PBX") system. PBXs are highly proprietary, expensive and relatively inflexible.

In a preferred embodiment of the present invention, the packet assembly circuit and the packet disassembly circuit are embodied in preprogrammed general purpose data processing and storage circuitry. Those of skill in the art will recognize that, while the system of the present invention may be embodied in discrete circuitry, microprocessor-based integrated circuits provide an attractive and flexible environment for embodiment of the system.

The foregoing has outlined rather broadly the features and technical advantages of the present invention so that those skilled in the art may better understand the detailed description of the invention that follows. Additional features and advantages of the invention will be described hereinafter that form the subject of the claims of the invention. Those skilled in the art should appreciate that they may readily use the conception and the specific embodiment disclosed as a basis for modifying or designing other structures for carrying out the same purposes of the present invention. Those skilled in the art should also realize that such equivalent assemblies do not depart from the spirit and scope of the invention in its broadest form.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

FIG. **1** illustrates a computer network that forms an environment within which the present invention can operate;

FIG. **2** illustrates a block diagram of a microprocessor-based system constructed in accordance with the present invention;

FIG. **3** illustrates an Ethernet data packet of audio data assembled according to the present invention;

FIG. **4** illustrates an ATM data cell of audio data assembled according to the present invention;

FIG. **5** illustrates the operation of the buffer of the present invention;

FIG. **6** illustrates a flow diagram of the method of assembling a data packet according to the present invention; and

FIG. **7** illustrates a flow diagram of the method of disassembling a data packet according to the present invention.

DETAILED DESCRIPTION

Referring initially to FIG. **1**, illustrated is a computer network, generally designated **100**, that forms an environment within which the present invention can operate. The network **100** is illustrated as including a telephone instrument **110** coupled, via a PC **120** having a display screen **124**, to an Ethernet-type computer network backbone **130**. Other telephone instruments **112**, **114** may be coupled to the backbone **130** via a multiple station card **122**. The present invention is capable of transmitting audio signals among the telephone instruments **110**, **112**, **114** via the Ethernet backbone **130**.

The present invention is compatible with various physical layer protocols. The Ethernet backbone **130** is linked through an Ethernet Switch **140** and an ATM hub **150** to a Token Ring backbone **172** of a Token Ring LAN **170**. The Token Ring backbone **172** is coupled, via a PC **176** having a display screen **178**, to a telephone instrument **174**. The ATM hub **150** is coupled, via a PC **154** to a display screen **156**, to a telephone instrument **154**. Packetized computer data transmitted across the Ethernet backbone **130** is switched through the Ethernet switch **140** to the ATM hub **150**. Packetized computer data transmitted across the Token Ring backbone **172** is routed directly through the ATM hub **150**. Again, the present invention is fully ATM-compatible, thereby allowing full access to ATM resources via the ATM hub **150**.

A telephone server **160** is connected to a plurality of telephone instruments **162**, **164** and connected, via the Ethernet Switch **140**, to the Ethernet backbone **130**. The telephone server **160** is also connected through the ATM hub **150**. Audio data from the Ethernet backbone **130** is directed through the telephone server **160**, via the Ethernet switch **140**, to the ATM hub **150**. The telephone server **160** provides full ISDN communication to central office ("CO") trunk lines **166**, thereby allowing WAN via ATM.

Again, the present invention provides a system and method for communicating audio data in the packet-based computer network **100** wherein transmission of data packets through the computer network **100** requires variable periods

of transmission time. The present invention is designed to operate in a distributed architecture network **100** with components as herein described.

The telephone instruments **110, 112, 114, 162, 154, 164, 174** may be traditional analog instruments, but it is within the scope of the present invention that they be ISDN-compatible or other digital instruments. The PCs **120, 154, 174** are illustrated as being conventional PCs having an expansion or input/output ("I/O") bus preferably adhering to the Industry Standard Architecture ("ISA") or Extended Industry-Standard Architecture ("EISA"). Those of skill in the art will understand that the present invention is not limited to a particular hardware architecture. As will be described with reference to FIG. **2**, the I/O bus provides an interface by which the system of the present invention allows communication between the backbones **130, 170** and the hub **150** and the corresponding PCs **120, 154, 174**.

The PC **120** includes a display screen **124** that is capable of displaying, under software control, data pertaining to operation of the system. This allows a user to use the display screen **124** for visual access to phone features through processing and interface capabilities, such as those provided in Telephony Application Programmers Interface ("TAPI"), developed by Intel and Microsoft or Telephony Services Application Programmers Interface ("TSAPI"), developed by Novell and AT&T. The backbone **130** is a conventional Ethernet backbone comprising multiple parallel conductors that act as paths along which data are transferred among nodes of the computer network **100**.

The ATM hub **150** is an interface card that converts Ethernet or Token Ring packet formats to ATM cell formats. The Ethernet packet to ATM cell conversion is discussed in reference to FIG. **4**. The ATM hub **150** provides the previously-described interface between the Ethernet or Token Ring network and an ATM-switched network.

In the illustrated embodiment, the telephone server **160** multiplexes signals from dedicated telephones **162, 164** and audio data from the backbone **130** of the Ethernet physical protocol layer, thereby providing digital service of audio data.

Turning now to FIG. **2**, illustrated is a block diagram of a microprocessor-based system constructed in accordance with the present invention. The microprocessor-based controller comprises a microprocessor **210**, a digital signal processor ("DSP") **220**, a CODEC **230**, a telephone set interface ("TSI") **240**, a TSI connector **242**, random-access memory ("RAM") **250**, an Ethernet controller **260**, an Ethernet controller interface connector **262**, a dual port memory **270**, and a dual port memory interface connector **272**.

The illustrated embodiment provides standard telephone instrument **110** connectivity into the PC **120** through the TSI **240** and TSI connector **242**. The TSI **240** accepts an analog signal from the telephone instrument **110**. The TSI connector **242** is preferably a standard RJ-11 connector.

The illustrated embodiment also provides connectivity to the backbone **130** through the Ethernet controller **260** and Ethernet controller interface connector **262**. The Ethernet controller **260** transmits data to, and receives data from, the backbone **130**. The Ethernet controller interface connector **262** is preferably a standard RJ-45 connector. The Ethernet controller **260** is internally connected to the processor **210** and RAM **250** by an internal local bus **265**.

The TSI **240** is coupled to the CODEC **230**. The CODEC **230** provides the analog-to-digital and digital-to-analog conversion for the audio data. The CODEC **230** comprises a digital conversion/compression circuit for digitizing and compressing the audio signal into the stream of digital audio data. Those of ordinary skill in the art should understand that the present invention does not depend upon application of a particular compression/decompression algorithm, or upon whether the data are even compressed at all. The sampling and compression schemes described herein are for illustration only.

When the telephone instrument **110** transmits an analog audio signal to the CODEC **230**, the CODEC **230** samples the signal at a predetermined, conventional rate of 8kHz. The CODEC **230** then preferably employs a known, standard logarithmic compression method (such as A-Law or μ-Law) to compress a 13 or 14 bit wide data sample into an 8 bit compressed sample. The CODEC **230** further comprises a decompression/analog conversion circuit for decompressing and converting the stream of digital audio data back into the audio signal. The decompression circuit restores the 8 bit compressed sample into a decompressed 13 or 14 bit sample and converts the sample into an analog voltage for reproduction in the telephone instrument **110**. Finally, the CODEC **230** has an associated clock (not illustrated) that governs the pace of the CODEC's operation.

The DSP **220** analyzes, filters and enhances audio data from the CODEC **230**. The DSP **220** may also provide echo cancellation or compression/decompression in lieu of the CODEC **230**. Echo cancellation is typically a requirement when the round trip audio delay exceeds 60 ms.

The processor **210** is charged with the responsibility of compiling the information from the DSP **220** and Ethernet controller **260** and performing the operations required to transmit the data. The processor **210** therefore embodies the packet assembly circuit and the packet disassembly circuit. As stated above, the packet assembly circuit generates a position identifier **370** that indicates a temporal position of the portion relative to the stream, inserts the position identifier **370** into the data packet and queues the data packet in the Ethernet controller for transmission through the Ethernet backbone **130**.

The RAM **250** preferably contains a receiving buffer **510** according to the present invention. It will be recalled that the receiving buffer **510** is associated with the packet disassembly circuit and provides the environment within which portions of audio data are reassembled.

The processor **210** further embodies an interpolation circuit for inserting synthesized audio data into a designated location of the receiving buffer **510** to thereby lengthen the portions of the stream of audio data in the receiving buffer **510** and a decimation circuit for deleting audio data from a designated location of the receiving buffer **510** to thereby shorten the portions of the stream of audio data in the receiving buffer **510**.

Access between the dual port memory **270** and the I/O bus **280** of the PC **120** is provided through the dual port memory connector **272**. The dual port memory **270** provides storage capacity and overflow back-up in facilitating communication between the internal local bus **265** and the I/O bus **280**. Digital data from the Ethernet controller **260** and the processor **210** can be stored in the dual port memory **270**.

At this point, it should be stated that the present invention is ultimately directed to application in an ATM environment. It has been stated previously that ATM does not currently enjoy wide acceptance. However, this is changing. Thus, with respect to the embodiments disclosed herein, a two-part description will be undertaken. In FIG. **3**, the present invention will be described as applied in the currently-popular

Ethernet environment. In FIG. 4, the present invention will be described as applied in ATM, its eventual preferred environment.

Turning now to FIG. 3, illustrated is an Ethernet data packet of audio data assembled according to the present invention. The preferred embodiment demonstrates the compatibility of the present invention with an Ethernet II frame having a total length of 74 bytes. A total frame size of 72 bytes is the minimum sized frame allowed by Ethernet. Illustrated are an Ethernet II header **310**, a message **330** and an Ethernet II trailer **390**.

The Ethernet II header **310** comprises an Ethernet preamble **313**, an Ethernet Start Frame Delimiter ("SFD") **316**, a destination address **319**, a source address **322** and a type field **325**. The Ethernet preamble **313** is a 7 byte series that provides timing synchronization for the receivers. The Ethernet SFD **316** is a 1 byte address that separates data at the input of the computer. The type field **325** denotes the upper-layer protocol that is using the data packet.

The Ethernet II header **310** further comprises the destination address **319** and source address **321** for determining a transmission route of the data packet through the computer network. Prior to transmitting the audio data across the backbone **130** of the computer network **100** of FIG. **1**, the data packet is assigned the destination address **319** and source address **322**. Each individual node in the computer network is designated by a specific address. To ensure that each individual data packet is routed to the proper destination, the Ethernet II header **310** of each data packet is assigned a respective destination address **319** and source address **322**. Consequently, the data travels between respective locations.

In particular, the destination address **319** marks the destination field that the data packet will be sent in the computer network. The source address **322** is the address of the station in the computer network that sent the data packet. Both the destination address **319** and the source address **322** are 6 bytes long.

The Ethernet II trailer **390** comprises a Frame Check Sequence ("FCS") field **395**. The FCS field **395** is an error-checking device built into each data packet to ensure that only valid frames are processed by the receiving station. The FCS field **395** contains a 4 byte CRC value. A CRC validation is performed by the transmitting stations before sending the data packet. The receiving station performs the same CRC validation, matching the resulting value against the contents of the FCS field. If the numbers match, the data packet is assumed to be valid, if not, the packet is disregarded.

The message **330** of the data packet has a maximum length of **48** bytes. The message **330** is comprised of a reserved/length field **340** (optional, and employed with variable-length audio data packets), a reserved/CRC field **350**, a channel identifier **360**, a position identifier **370**, and a audio data sample **380**.

The reserved/length field **340** is **1** byte long and specifies the number of bytes contained between the reserved/length field **340** and the last byte in the audio data sample **380**. The reserved/CRC field **350** is a 1 byte field reserved for error checking purposes in an ATM cell. The channel identifier **360** is a 1 byte field that identifies the message **330** as a packet of control data (perhaps containing signalling commands) if the channel identifier **360** is equal to **255** otherwise it represents the audio data of a specific station. The channel identifier allows multiple voice connections on a single real channel to save switching complexity within the data network. It also allows voice conferencing on shared media without additional dedicated bandwidth. The channel identifier is also used in a call setup sequence to allow multiple conversations between two voice server devices, thereby suitable for ATM transport.

The position identifier **370** is a pointer representing the newest audio sample **380**. The position identifier **370** is a 1 byte long pointer to 4 byte words of the audio sample **380** and can represent 256×4 bytes (1 kilobyte) before it overflows and wraps. Since digitized audio typically uses a standard 8kHz sampling rate (125 microseconds between samples), 256×4×125 microseconds is the total time that the position identifier **370** can represent before wrapping. The position identifier **370** is used both when the channel identifier **360** represents audio data and when the channel identifier **360** represents control data (such as signalling or call processing). For example, when the channel identifier **360** equals **255** then the position identifier **370** is used to represent a signalling data message type.

Finally, the message **330** of the data packet contains up to **44** bytes of digitized audio data samples **380**. The audio data samples **380** contain digitized audio data if the channel identifier **360** is a value other than "255." The audio data sample **380** contains system commands if the channel identifier equals "255." The commands may be, for example, information blocks used to set up, take down, forward and conference telephone calls.

The present invention is designed to handle data packets of variable-size, to manage variable time transmission of data and to increase the throughput efficiency of data across the backbone **130** of the computer network. This attribute is extremely important to transmitting time-sensitive audio data to achieve high audio fidelity.

Turning now to FIG. 4, illustrated is an ATM data cell of audio data assembled according to the present invention. The preferred embodiment demonstrates the compatibility of the present invention with an ATM cell having a total, fixed length of 53 bytes. The cell is characterized by an ATM header **410** preceding a message (the message **330** of FIG. **3**).

ATM combines the benefits of both circuit switching and cell switching by providing multiple switched virtual circuit connections to users through a single access to a network. The ATM header **410** contains information specifying the virtual path (a Virtual Path Identifier ("VPI") **430**) and virtual channel (Virtual Channel Identifier ("VCI") **440**) of the cell. The VPI **430** and VCI **440** together establish a node-to-node communications channel. Switch routing is based on the VPI **430** and VCI **440**. The ATM switch requires a connection to be established between the incoming and outgoing virtual channels before information can be routed through the switch. The ATM switch then switches and routes each individual cell from the incoming multiplexed cell stream to the outgoing multiplexed cell stream based upon the virtual channels identified within the ATM header **410**. In this context, ATM is truly seen as a connection-oriented technology. The ATM switch maintains cell sequence; and each cell is switched at the cell rate, not the channel rate, to accommodate for variable bit rate transmissions.

A Cell Loss Priority Field ("CLP") **460** within the ATM header **410** establishes priority on the network. There are two levels of semantic priority that allows users or network providers to choose which cells to discard during periods of network congestion. The types are defined by a "1" or "0" in the CLP **460** within the ATM header **410**. During periods

of congestion, the CLP **460** determines which information will be discarded or switched through the network.

The Payload Type Indicator ("PTI") **450** in the ATM header **410** discriminates between a cell carrying user information (such as audio data) or service information (such as control data) in the message field **330**. The Header Error Control field ("HEC") **470** provides error checking of the ATM header **410**.

The Generic Flow Control field ("GFC") **420** of the ATM header **410** is designed to provide shared public access similar to the functionality of a Metropolitan Area Network ("MAN"). GFC **420** is used when there is a single user access point servicing multiple terminal interfaces, such as those found in a LAN environment. Each terminal must receive equal access to the network facilities, and the GFC **420** ensures that each terminal will get equal access to the shared network bandwidth. The GFC **420** will manage the various LAN topologies and architectures.

The six fields are positioned within the 5 byte ATM header **410** at address locations as displayed in the illustrated embodiment. Distinct from an Ethernet data packet, the ATM cell transmits information through the network intact with no error checking or correction performed on the message field **330**. The reserved/CRC field **350** is reserved to perform error checking on the channel identifier **360**, the position identifier **370** and the audio data sample **380** in an ATM cell at the receiving end. The message field **330** and contents therein are as described in relation to the corresponding portions of the Ethernet data packet previously described in conjunction with FIG. **3**. Translation between an Ethernet data packet and an ATM cell is completed by stripping the destination address **319** and source address **321** from the message field **330** and converting the source and destination addresses **319**, **321** to the VPI **430**, VCI **440** and channel identifier **360** associated with the ATM cell.

Turning now to FIG. **5**, illustrated is the operation of the receiving buffer **510** of the present invention. As previously discussed, the system is comprised of a packet disassembly circuit, having the receiving buffer **510** located in the RAM **250** associated therewith, for receiving the audio data sample **380** from the backbone **130**. The packet disassembly circuit inserts the portion into an absolute location of the receiving buffer **510**, the position identifier **370** determining the location. The audio data sample **380** is thereby synchronized with adjacent audio data samples **380** in the receiving buffer **510** to compensate for the variable periods of transmission time. The CODEC reads from the receiving buffer, lagging the audio data samples, as they are inserted, by some period of time (20 ms in the illustrated embodiment), thereby creating a travelling window in the receiving buffer **510** of 20 ms delay. Since the receiving buffer is of a physical finite length (about 1 kilobyte in the preferred embodiment), the window "wraps around" the addresses of the receiving buffer **510**. Thus, at any given addressable location within the receiving buffer **510** data are first written to the location, then read from, then written to again, and so on. The receiving buffer **510** therefore acts as a fixed-delay playback buffer.

Again, in the illustrated embodiment, the length of the window in the receiving buffer **510** is about 20 ms. The window is software setable at that value to account for jitter in the transmission network, and packetizing and depacketizing delay. The jitter in the network is primarily due to data traffic congestion. The pre-set length of the window more than adequately accommodates a data packet and any inherent system delays in reconstructing the audio data at the receiving end.

As previously mentioned, the CODEC **230** reads from the receiving buffer **510** at a rate ideally equal to that at which audio data are added, thereby maintaining window length. As data are read, the data are replaced with white noise data, representing silence. If the white noise data are not subsequently overwritten with received audio data in a subsequent pass through the receiving buffer **510**, the CODEC **230** reads and decompresses the white noise data instead, producing a synthesized near-silence for the benefit of the listener in lieu of audio data.

FIG. **5** specifically illustrates **6** audio data samples **380** of various sizes and variable transmission delays being placed into the receiving buffer **510** as a function of the position identifier **370** contained in each data packet. A value of the position identifier **370** may be a function of a length of audio data sample **380** in a previously-transmitted data packet but is not constrained thereby. The position identifier **370** directs each audio data sample **380** into specified absolute positions of the receiving buffer **510** at the receiving end. Thus, the position identifier **370** is fundamentally different from a packet sequence number.

FIG. **5**, in conjunction with the following Table I, illustrates insertion of audio data samples into the receiving buffer **510** according to the present invention.

TABLE I

Audio Data Sample Number	Audio Data Sample Size (bytes)	Delay of each packet (ms)	Position Identifier	CODEC Read Offset	Buffer Length (ms)
1	44	0	29	0	20
2	44	1	40	13	19
3	44	10	51	42	10
4	44	4.5	62	42	15.5
5	24	1.5	73	42	18.5
6	44	0	79	50	20
7	44	0	90	60	20.5

Again, at a sample rate of 8kHz, individual bytes or samples occur in 0.125 ms intervals. "Position identifier" ("PI") locates each temporally successive audio data sample **380** in an absolute position within the receiving buffer **510**. The PI is divided by 4, such that a PI of **6** actually points to byte **24** in the receiving buffer **510**.

The "CODEC Read Offset" ("CRO") reflects the read position with respect to the CODEC in the receiving buffer **510**. Analogous to the PI, the CRO is the actual CODEC read position divided by 4, such that a CRO of 1 actually points to byte **4** in the receiving buffer **510**. In the illustrated embodiment, sample 1 contains 44 bytes of data without a delay in the system. Thus, CRO_1 is 0 and PI_1 is 29, resulting in a 20 ms buffer length (14.5 ms plus 5.5 ms of sample 1). The 44 bytes of audio data sample **380** are placed in the last 5.5 ms of the receiving buffer **510**.

In sample **2**, the system experiences a 1 ms delay. The 44 bytes of audio data sample **380** are placed adjacent to sample 1 with PI_2 equal to 40. Since the audio data sample **380** is delayed 1 ms, CRO_2 equals 13, equating to a total of 6.5 ms. Thus, the difference between PI_2 and CRO_2 contracts to a 27 position difference. Adding the 27 position difference between PI_2 and CRO_2 to the 44 bytes of audio data sample **380** equates to a 19 ms window for sample 2. A 10 ms system delay is encountered by sample 3, leading to a contraction of the window to 10 ms. In samples 4 and 5, the system has compensated for some of the delay and, as a result, the length of the windows has increased as shown. As

previously discussed, the position identifier **370** represents an absolute position in the receiving buffer **510** regardless of the delay in the system. Furthermore, once the transport media is free after the extended delay associated with sample 3, samples 3–5 are immediately positioned in the receiving buffer **510** one after the other as shown.

Sample 5 further illustrates the circumstance when a shortened audio data sample **380** is transmitted. Sample 5, which is only 24 bytes long, is inserted into the receiving buffer at PI_5=73. Since sample 5 is short by 20 bytes, the missing 20 bytes are filled with white noise, representing silence. The silence is not shown, as will be explained.

Next sample **6** arrives. Sample 6 is a full-length packet of 44 bytes. Thus, PI_6 equals 79. Sample 6 overwrites the 20 bytes of silence that had been appended to the end of sample 5. Since FIG. **5** already shows sample 6 in place, the silence is already overwritten and thus not shown.

Finally, sample 7 displays the circumstance when the CODEC clock operates too slowly. For purposes of discussion, the CODEC clock is assumed to be grossly out of frequency, such that the effect produced thereby is emphasized. In such case, PI advances 5.5 ms or **11** positions from the previous PI to position **90** in the receiving buffer **510**. However, the slow CODEC clock forces the CRO to lag. In this instance, the CRO only advances 5.0 ms or **10** positions from the previous CRO to position **60** in the receiving buffer **510**. The result is that the length of the window is 20.5 ms. Decimation is therefore required to shorten the receiving buffer **510** to the pre-set size.

Decimation is performed in adjustment intervals as follows: 1 byte for every 2 bytes away from the ideal window length (**160** bytes, in the illustrated embodiment), 2 bytes for every 3 or 4 bytes away from the ideal window length and 3 bytes for every 5 or 6 bytes away from the ideal window length. In this instance, the buffer is 0.5 ms too long, equating to 4 bytes. Accordingly, the decimation circuit must remove 2 bytes from the receiving buffer **510** to adjust the receiving buffer **510** window toward the ideal length. Interpolation and decimation are ongoing processes in the system of the present invention.

Before leaving FIG. **5**, it should be noted that, if window length is reduced to zero (either by virtue of the non-transmission of periods of silence or by virtue of reception of multiple invalid packets), the CODEC **230** simply reads the white noise in the receiving buffer **510**, thereby simulating silence, again for the benefit of the listener.

Turning now to FIG. **6**, illustrated is a flow diagram of the method of assembling a data packet according to the present invention. The packet assembly circuit constructs a data packet from a portion of a stream of digital audio data corresponding to an audio signal. As illustrated in the preferred embodiment, in a step **610**, a sample of audio data are received into the packet assembly circuit. In a decisional step **620**, the packet assembly circuit determines whether the sample represents silence or nonsilence by comparing the data therein to a predetermined threshold. If the data have a value less than the threshold, a packet is not generated, as it is of little value to occupy network bandwidth transmitting silence. If the data have a value equalling or exceeding the threshold, execution proceeds to a step **630**, wherein the packet assembly circuit assigns the reserved/length field **340**, the reserved/CRC field **350**, the channel identifier **360** and the position identifier **370** to the audio data sample **380**. The previously-described fields appended to the audio data sample **380** constitute the message **330**.

In a step **640** (only applicable in an Ethernet environment), the Ethernet II header **310** and Ethernet II trailer **390** are affixed to the message **310**. The Ethernet II header **310** and Ethernet II trailer **390** contain information necessary to route the data packet through the computer network and to check the transmitted data for errors. In an ATM environment, an ATM header is affixed to the packet.

In a step **650** (again, only applicable in an Ethernet environment), the data packet is evaluated for errors. If there is an error in the data packet, the process restarts, otherwise the process moves to a step **660**. In the step **660**, the data packet is queued for transmission across the backbone of the network.

Turning now to FIG. **7**, illustrated is a flow diagram of the method of disassembling a data packet according to the present invention. In a step **710**, if the receiver accepts an invalid packet, the packet is disregarded and the disassembling process for that packet terminates in a step **720**.

In a step **730**, assuming the packet is valid, the packet disassembly circuit strips the reserved/length field **340**, the reserved/CRC field **350**, the channel identifier **360** and the position identifier **370** from the audio data sample **380**. In an Ethernet environment, the packet disassembly circuit also strips the Ethernet II header **310** and Ethernet II trailer **390**.

In a step **740**, the packet disassembly circuit inserts the audio data sample **380** into an absolute location of the receiving buffer **510** (of FIG. **5**) according to the value of the position identifier **370**. The audio data sample **380** is thereby synchronized with adjacent audio data samples **380** of the stream of digital audio data in the receiving buffer **510** to compensate for the variable periods of transmission time.

From the above, it is apparent that the present invention provides a system and method for communicating audio data in a packet-based computer network wherein transmission of data packets through the computer network requires variable periods of transmission time. The system comprises: (1) a packet assembly circuit for constructing a data packet from a portion of a stream of digital audio data corresponding to an audio signal, the packet assembly circuit generating a position identifier indicating a temporal position of the portion relative to the stream, inserting the position identifier into the data packet and queuing the data packet for transmission through a backbone of the computer network and (2) a packet disassembly circuit, having a buffer associated therewith, for receiving the data packet from the backbone, the packet disassembly circuit inserting the portion into an absolute location of the buffer, the position identifier determining the location, the portion thereby synchronized with adjacent portions of the stream of digital audio data in the buffer to compensate for the variable periods of transmission time.

Although the present invention and its advantages have been described in detail, those skilled in the art should understand that they can make various changes, substitutions and alterations herein without departing from the spirit and scope of the invention in its broadest form.

What is claimed is:

1. A system for communicating audio data in a packet-based computer network, transmission of data packets through said computer network requiring variable periods of transmission time, the system comprising:

a packet assembly circuit for constructing a data packet from a portion of a stream of digital audio data corresponding to an audio signal, said packet assembly circuit generating a position identifier indicating a temporal position of said portion relative to said stream, inserting said position identifier into said data packet and queuing said data packet for transmission through a backbone of said computer network; and

a packet disassembly circuit, having a buffer associated therewith, for receiving said data packet from said backbone, said packet disassembly circuit inserting said portion into an absolute location of said buffer, said position identifier determining said location, said portion synchronized with adjacent portions of said stream of digital audio data in said buffer to compensate for said variable periods of transmission time.

2. The system as recited in claim 1 further comprising an interpolation circuit for inserting synthesized audio data into a designated location of said buffer to lengthen said portions of said stream of audio data in said buffer.

3. The system as recited in claim 1 further comprising a decimation circuit for deleting audio data from a designated location of said buffer to shorten said portions of said stream of audio data in said buffer.

4. The system as recited in claim 1 wherein said data packet comprises source and destination fields for determining a transmission route of said data packet through said computer network.

5. The system as recited in claim 1 wherein a value of said position identifier is a function of a length of a portion of said stream of digital audio data in a previously-transmitted data packet.

6. The system as recited in claim 1 wherein a window of said buffer is about 20 milliseconds.

7. The system as recited in claim 1 wherein said data packet is capable of containing a portion having a length of about 5.5 milliseconds.

8. The system as recited in claim 1 further comprising a digital conversion/compression circuit, coupled to said packet assembly circuit, for digitizing and compressing said audio signal into said stream of digital audio data.

9. The system as recited in claim 1 further comprising a decompression/analog conversion circuit, coupled to said packet disassembly circuit, for decompressing and converting said stream of digital audio data back into said audio signal.

10. The system as recited in claim 1 wherein said computer network comprises a plurality of computers coupled to said backbone, said packet assembly circuit and said packet disassembly circuit located in separate ones of said computers.

11. A method of communicating audio data in a packet-based computer network, transmission of data packets through said computer network requiring variable periods of transmission time, the method comprising the steps of:

constructing a data packet from a portion of a stream of digital audio data corresponding to an audio signal with a packet assembly circuit, said packet assembly circuit generating a position identifier indicating a temporal position of said portion relative to said stream, inserting said position identifier into said data packet and queuing said data packet for transmission through a backbone of said computer network; and

receiving said data packet from said backbone into a packet disassembly circuit having a buffer associated therewith, said packet disassembly circuit inserting said portion into an absolute location of said buffer, said position identifier determining said location, said portion synchronized with adjacent portions of said stream of digital audio data in said buffer to compensate for said variable periods of transmission time.

12. The method as recited in claim 11 further comprising the step of inserting synthesized audio data into a designated location of said buffer to lengthen said portions of said stream of audio data in said buffer.

13. The method as recited in claim 11 further comprising the step of deleting audio data from a designated location of said buffer to shorten said portions of said stream of audio data in said buffer.

14. The method as recited in claim 11 further comprising the step of determining a transmission route of said data packet through said computer network with source and destination fields in said data packet.

15. The method as recited in claim 11 further comprising the step of assigning a value of said position identifier as a function of a length of a portion of said stream of digital audio data in a previously-transmitted data packet.

16. The method as recited in claim 11 further comprising the step of establishing a window of said buffer at about 20 milliseconds.

17. The method as recited in claim 11 further comprising the step of containing a portion having a length of about 5.5 milliseconds in said data packet.

18. The method as recited in claim 11 further comprising the step of digitizing and compressing said audio signal into said stream of digital audio data with a digital conversion/compression circuit coupled to said packet assembly circuit.

19. The method as recited in claim 11 further comprising the step of decompressing and converting said stream of digital audio data back into said audio signal with a decompression/analog conversion circuit coupled to said packet disassembly circuit.

20. The method as recited in claim 11 wherein said computer network comprises a plurality of computers coupled to said backbone, said method further comprising the step of locating said packet assembly circuit and said packet disassembly circuit in separate ones of said computers.

21. A packet-based computer network, comprising:

a backbone coupling, and for communicating packeted data between, first and second computer nodes, serial transmission of data packets through said computer network requiring variable periods of transmission time;

means, coupled to said first node, for receiving an original audio signal and generating therefrom a corresponding stream of digital audio data;

a packet assembly circuit, associated with said first computer node, for constructing data packets from portions of said stream of digital audio data, each of said data packets including:

one of said portions, and

a position identifier indicating a temporal position of said one of said portions relative to said stream, said packet assembly circuit queuing said data packet for serial transmission to said second node through said backbone;

a packet disassembly circuit, associated with said second computer node and a buffer, for serially receiving said data packets from said backbone, said packet disassembly circuit disassembling each of said data packets by:

inserting said portion into an absolute location of said buffer, said position identifier determining said location, said portion synchronized with adjacent portions of said stream of digital audio data in said buffer to compensate for said variable periods of transmission time; and

means, coupled to said second node, for generating a reconstructed audio signal from said stream of digital audio data in said buffer.

22. The network as recited in claim 21 further comprising an interpolation circuit for inserting synthesized audio data

into a designated location of said buffer to lengthen said portions of said stream of audio data in said buffer.

23. The network as recited in claim 21 further comprising a decimation circuit for deleting audio data from a designated location of said buffer to shorten said portions of said stream of audio data in said buffer.

24. The network as recited in claim 21 wherein said data packet comprises source and destination fields for designating said first node as a source and said second node as a destination of said data packet.

25. The network as recited in claim 21 wherein a value of said position identifier is a function of a length of a portion of said stream of digital audio data in a previously-transmitted data packet.

26. The network as recited in claim 21 wherein a window of said buffer is about 20 milliseconds.

27. The network as recited in claim 21 wherein said data packet is capable of containing a portion having a length of about 5.5 milliseconds.

28. The network as recited in claim 21 wherein said receiving means comprises a digital conversion/compression circuit, coupled to said packet assembly circuit, for digitizing and compressing said audio signal into said stream of digital audio data.

29. The network as recited in claim 21 wherein said generating means comprises a decompression/analog conversion circuit, coupled to said packet disassembly circuit, for decompressing and converting said stream of digital audio data back into said audio signal.

30. The network as recited in claim 21 wherein said packet assembly circuit and said packet disassembly circuit are embodied in preprogrammed general purpose data processing and storage circuitry.

31. A method for communicating packeted data over a backbone coupling first and second computer nodes of a packet-based computer network, serial transmission of data packets through said computer network requiring variable periods or transmission time, said method comprising the steps of:

receiving an original audio signal at said first computer node and generating therefrom a corresponding stream of digital audio data;

constructing data packets from portions of said stream of digital audio data with a packet assembly circuit associated with said first computer node, each of said data packets including:

one of said portions, and

a position identifier indicating a temporal position of said one of said portions relative to said stream,

said data packet queued for serial transmission to said second computer node through said backbone;

serially receiving said data packets from said backbone with a packet disassembly circuit associated with said second computer node and a buffer, said packet disassembly circuit disassembling each of said data packets by:

inserting said portion into an absolute location of said buffer, said position identifier determining said location, said portion synchronized with adjacent portions of said stream of digital audio data in said buffer to compensate for said variable periods of transmission time; and

generating a reconstructed audio signal from said stream of digital audio data in said buffer.

32. The method as recited in claim 31 further comprising the step of inserting synthesized audio data into a designated location of said buffer to lengthen said portions of said stream of audio data in said buffer.

33. The method as recited in claim 31 further comprising the step of deleting audio data from a designated location of said buffer to shorten said portions of said stream of audio data in said buffer.

34. The method as recited in claim 31 further comprising the step of designating said first node as a source and said second node as a destination of said data packet with source and destination fields in said data packet.

35. The method as recited in claim 31 further comprising the step of assigning a value of said position identifier as a function of a length of a portion of said stream of digital audio data in a previously-transmitted data packet.

36. The method as recited in claim 31 further comprising the step of establishing a window of said buffer at about 20 milliseconds.

37. The method as recited in claim 31 further comprising the step of containing a portion having a length of about 5.5 milliseconds.

38. The method as recited in claim 31 wherein said step of receiving comprises the step of digitizing and compressing said audio signal into said stream of digital audio data.

39. The method as recited in claim 31 wherein said step of generating comprises the step of decompressing and converting said stream of digital audio data back into said audio signal.

40. The method as recited in claim 31 further comprising the step of embodying said packet assembly circuit and said packet disassembly circuit in preprogrammed general purpose data processing and storage circuitry.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,526,353
DATED : June 11, 1996
INVENTOR(S) : Arthur Henley, et al.

It is certified that error appears in the above-indentified patent and that said Letters Patent is hereby corrected as shown below:

Column 19, line 38, "or" should be --of--.

Signed and Sealed this

Third Day of September, 1996

Attest:

BRUCE LEHMAN

Attesting Officer

Commissioner of Patents and Trademarks

US006259677B1

(12) **United States Patent**
Jain

(10) **Patent No.: US 6,259,677 B1**
(45) **Date of Patent: Jul. 10, 2001**

(54) **CLOCK SYNCHRONIZATION AND DYNAMIC JITTER MANAGEMENT FOR VOICE OVER IP AND REAL-TIME DATA**

(75) Inventor: **Jaswant R. Jain**, Chatsworth, CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/164,431**

(22) Filed: **Sep. 30, 1998**

(51) **Int. Cl.**[7] **H04J 3/24**
(52) **U.S. Cl.** **370/252**; 370/352; 370/509; 370/516
(58) **Field of Search** 370/389, 352, 370/356, 400, 401, 503, 507, 508, 509, 516, 517, 518, 519, 252; 375/356, 359, 371

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,530,091	*	7/1985	Crockett	370/503
4,894,823	*	1/1990	Adelmann et al.	370/252
5,255,291		10/1993	Holden et al.	375/111
5,790,548	*	8/1998	Sugar	370/352
6,072,809	*	6/2000	Agrawal et al.	370/503
6,101,195	*	8/2000	Lyons et al.	370/498
6,157,653	*	12/2000	Kline et al.	370/412

OTHER PUBLICATIONS

Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks by Ramachandran Ramjee, Jim Kurose, Don Towsley and Henning Schulzrinne, University of Massachusetts/GMD–Fokus, pp. 1–9.
Techniques for Packet Voice Synchronization by Warren Montgomery, Dec. 12, 1983, IEEE, pp. 1–7.
Adaption Algorithms, Playout Algorithms, RTP, Transport Protocols, Service Classes, Mapping Classes and Qos by Isidor Kouvelas, Jan. 27, 1997, GMT, 11 pages.
Analysis and Control of Audio Packet Loss over Packet–Switched Networks by Jean Chrysostome Bolot and Hugues Crepin, Sophia–Antipolis Cedex, undated, pp. 1–6.
Characterizing End–to–End Packet Delay and Loss in the Internet by Jean–Chrysostome Bolot, Sophia–Antipolis Cedex, Dec. 1993, Journal of High–Speed Networks, pp. 1–11.
A Distributed Experimental Communications System by John DeTreville and David W. Sincoskie, Dec. 1983, IEEE Journal on Selected Areas in Communications, pp. 1070–1075 (6 pages attached).

* cited by examiner

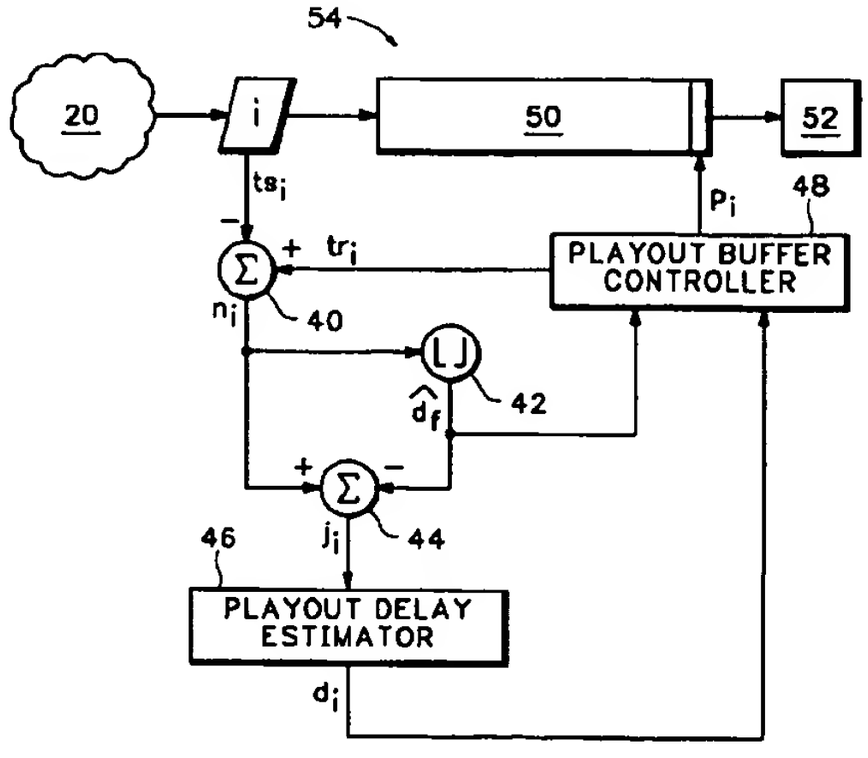
Primary Examiner—Ricky Ngo
(74) *Attorney, Agent, or Firm*—Marger Johnson & McCollom P.C.

(57) **ABSTRACT**

A real-time receiver and method for receiving and playing out real-time packetized data are disclosed. The receiver includes a packet transmission fixed delay estimator and a packet transmission variable delay estimator. The fixed delay estimator determines, using packets received up to the current point in a conference, the non-variable portion of observed delays. This non-variable portion is subtracted from each packet's observed delay to obtain a variable delay estimate for that packet.

Since variable delays actually drive the buffering time needed at the receiver to achieve smooth playout, the packet variable delay estimates can be used directly to adjust playout delay. Adaptive playout delay is preferably set aggressively low, based on observed packet variable delay estimates, to reduce data latency. Playout delay can be adjusted rapidly upwards when higher packet delays are observed, allowing rapid adaptation to network statistical variations and reducing the frequency of late packets.

18 Claims, 6 Drawing Sheets



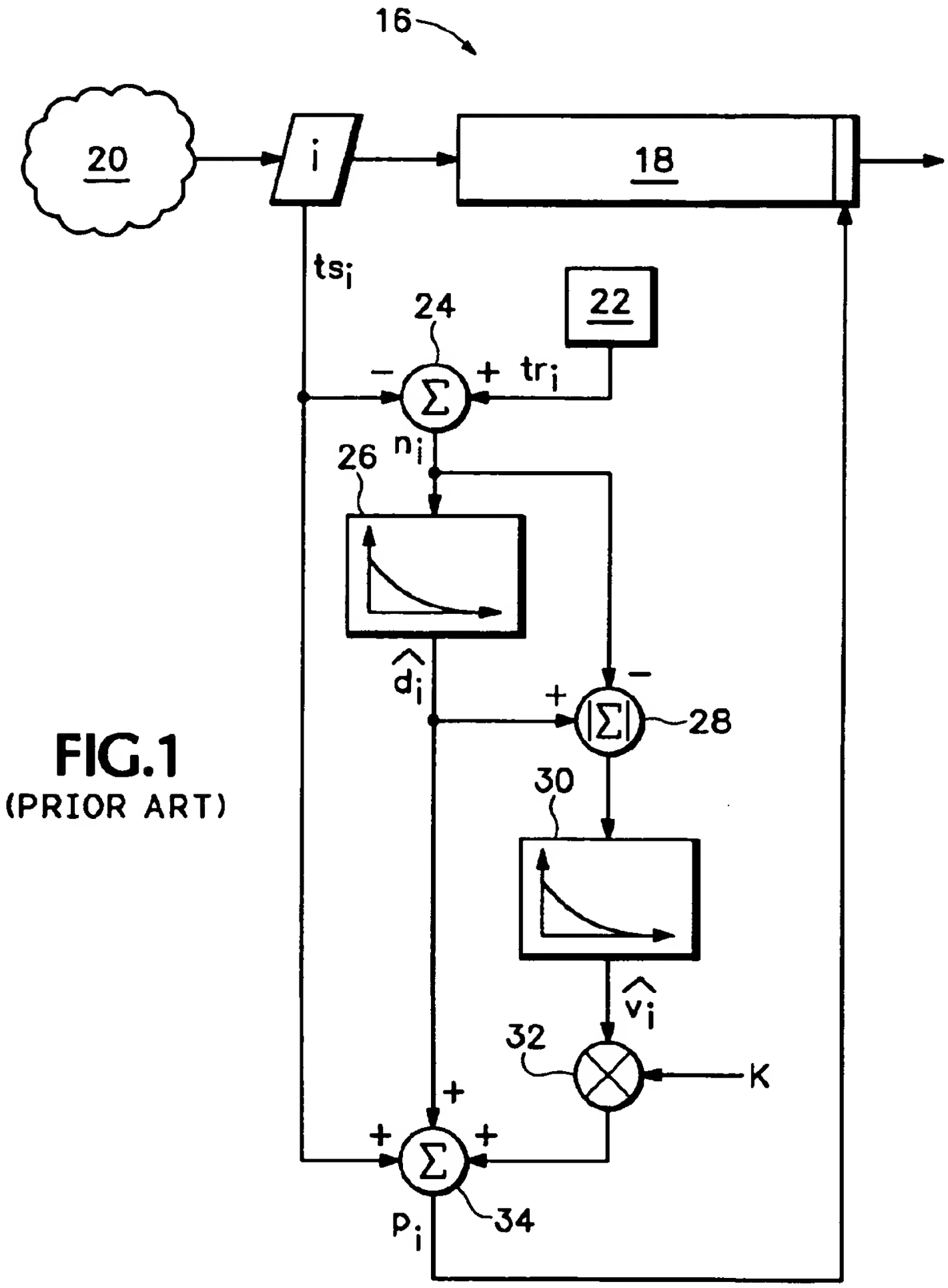


FIG.1
(PRIOR ART)

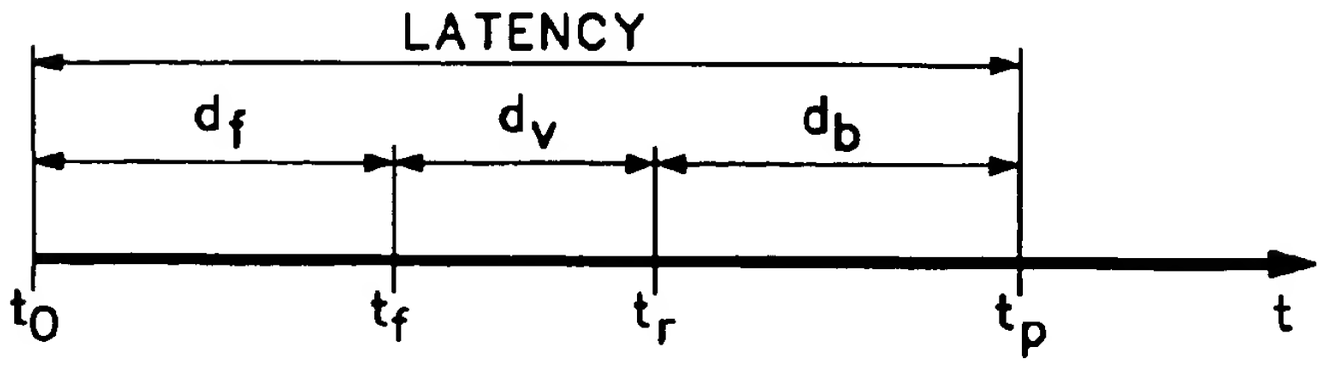


FIG.2

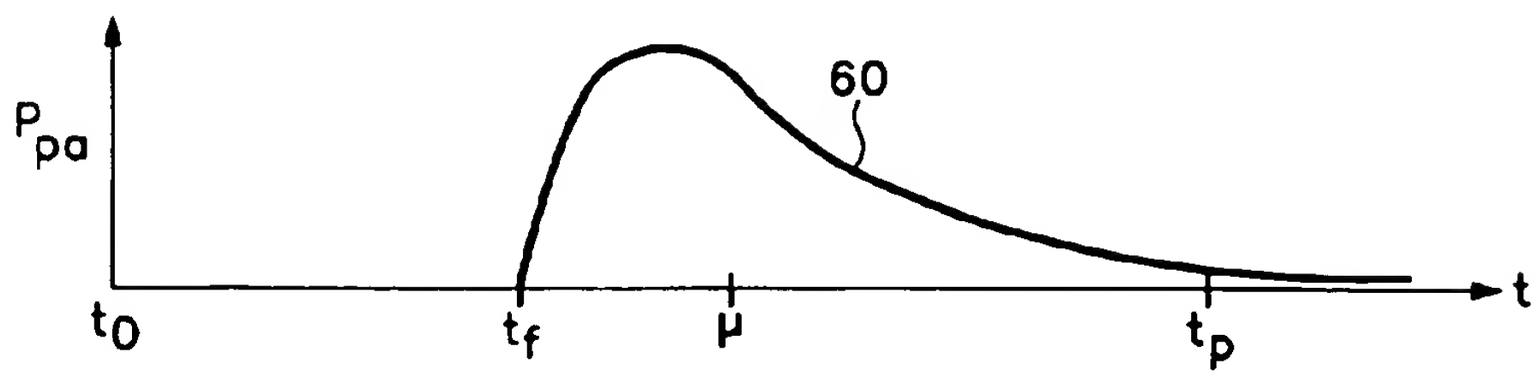


FIG.3

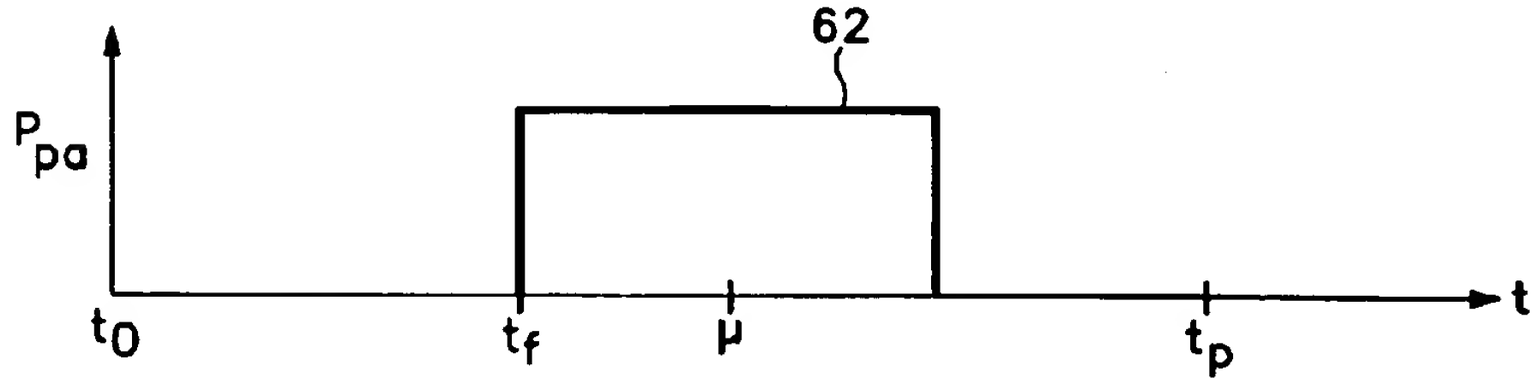


FIG.4

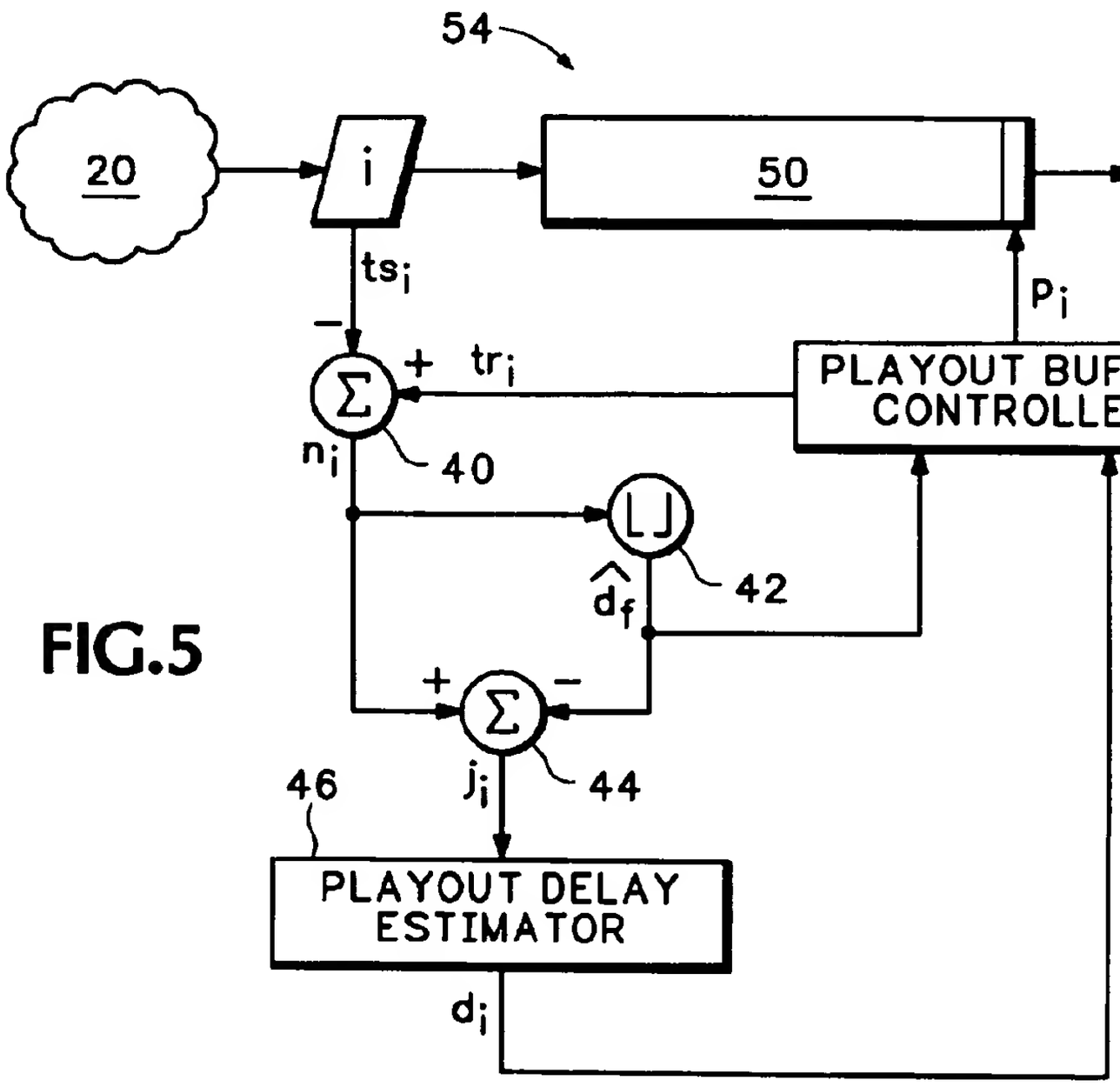


FIG.5

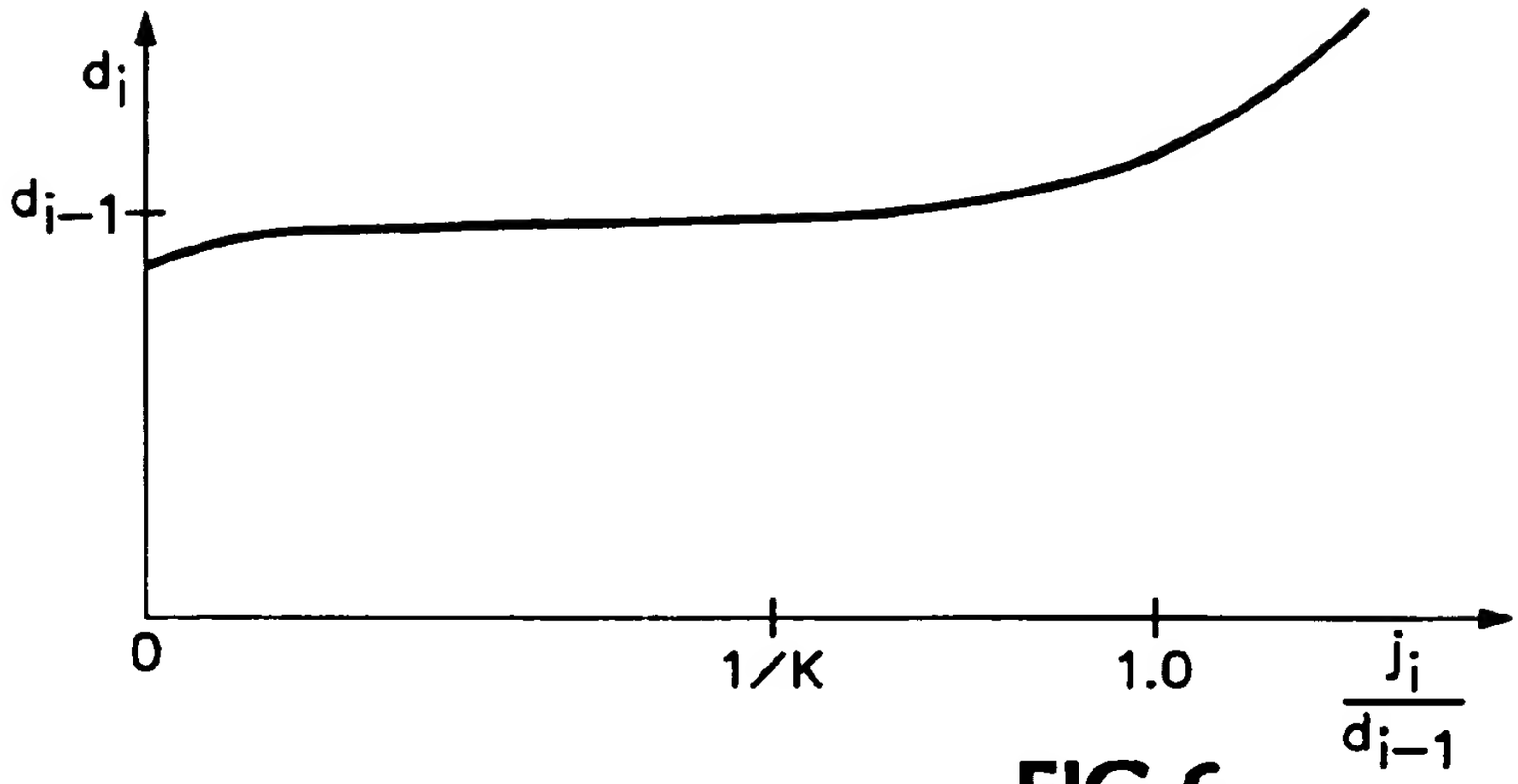


FIG.6

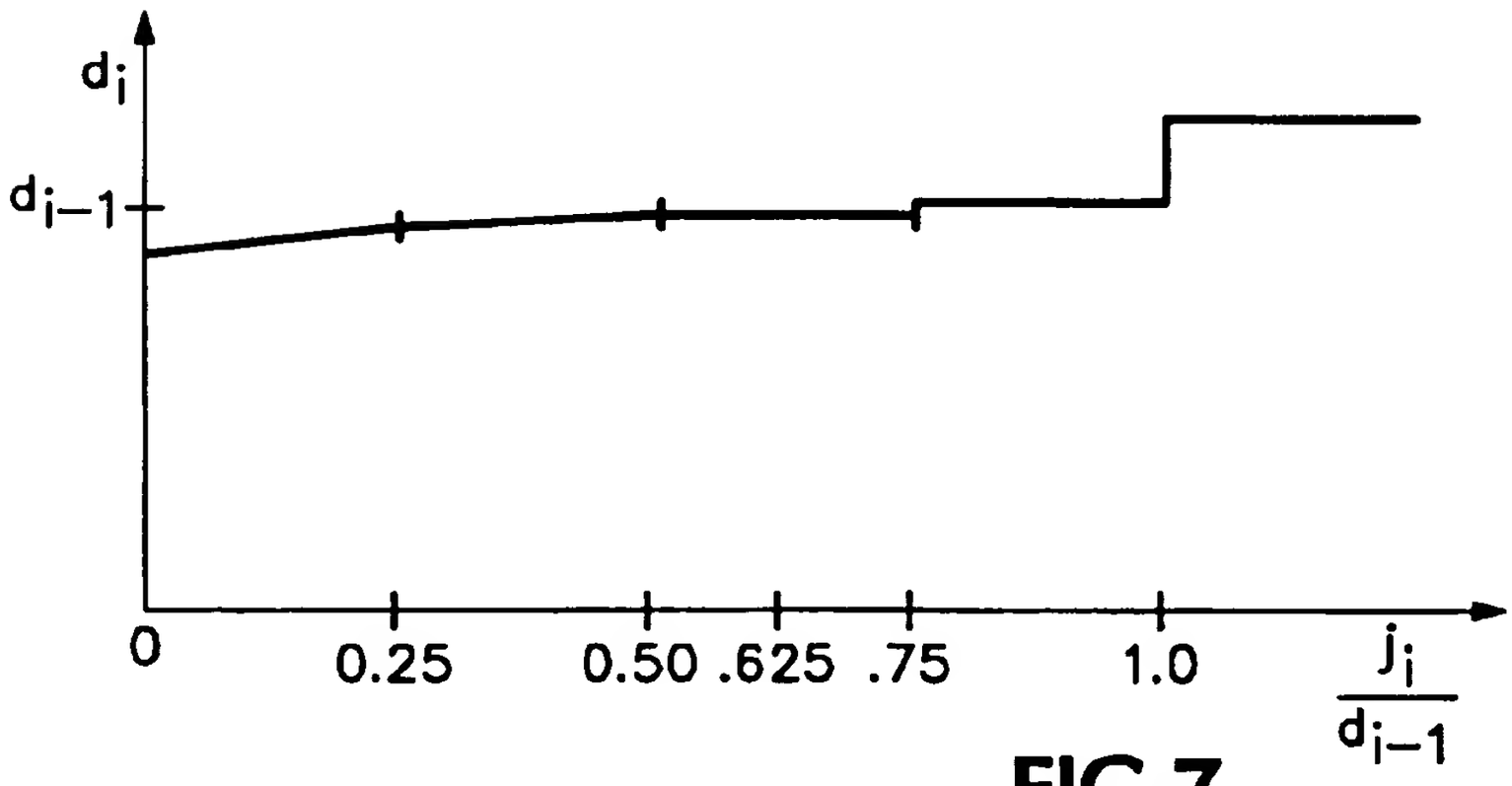


FIG.7

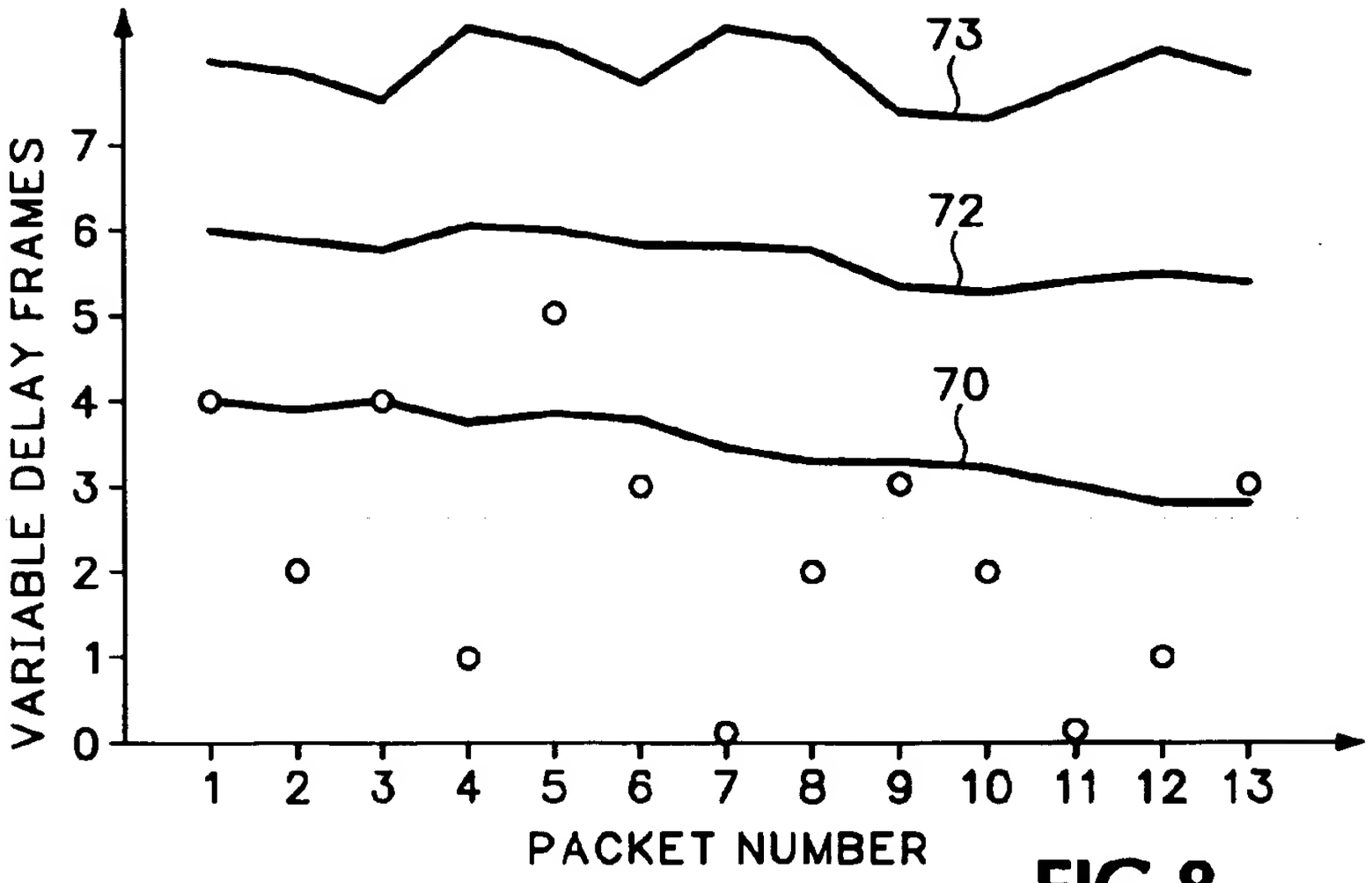


FIG.8
(PRIOR ART)

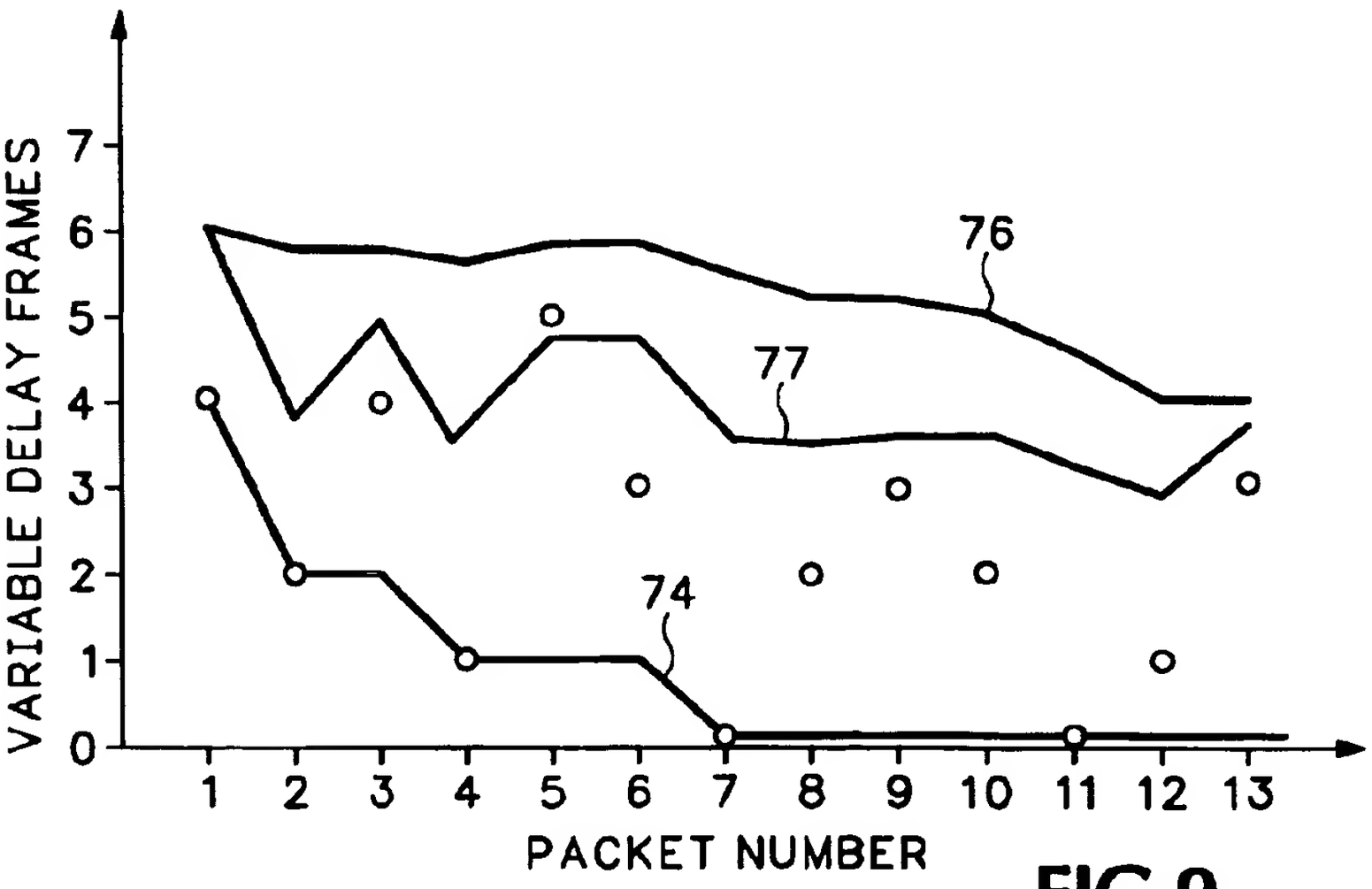


FIG.9

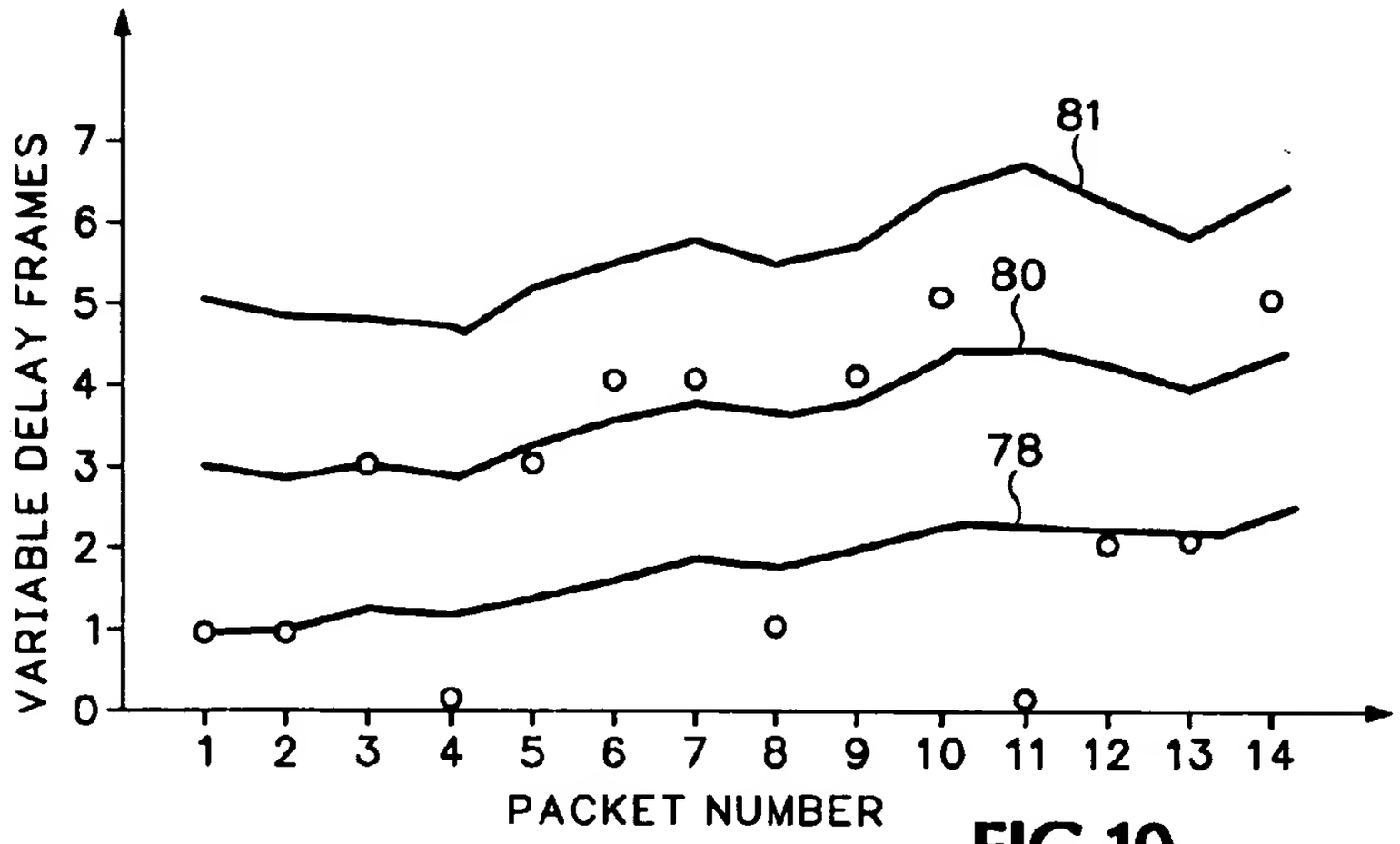


FIG.10
(PRIOR ART)

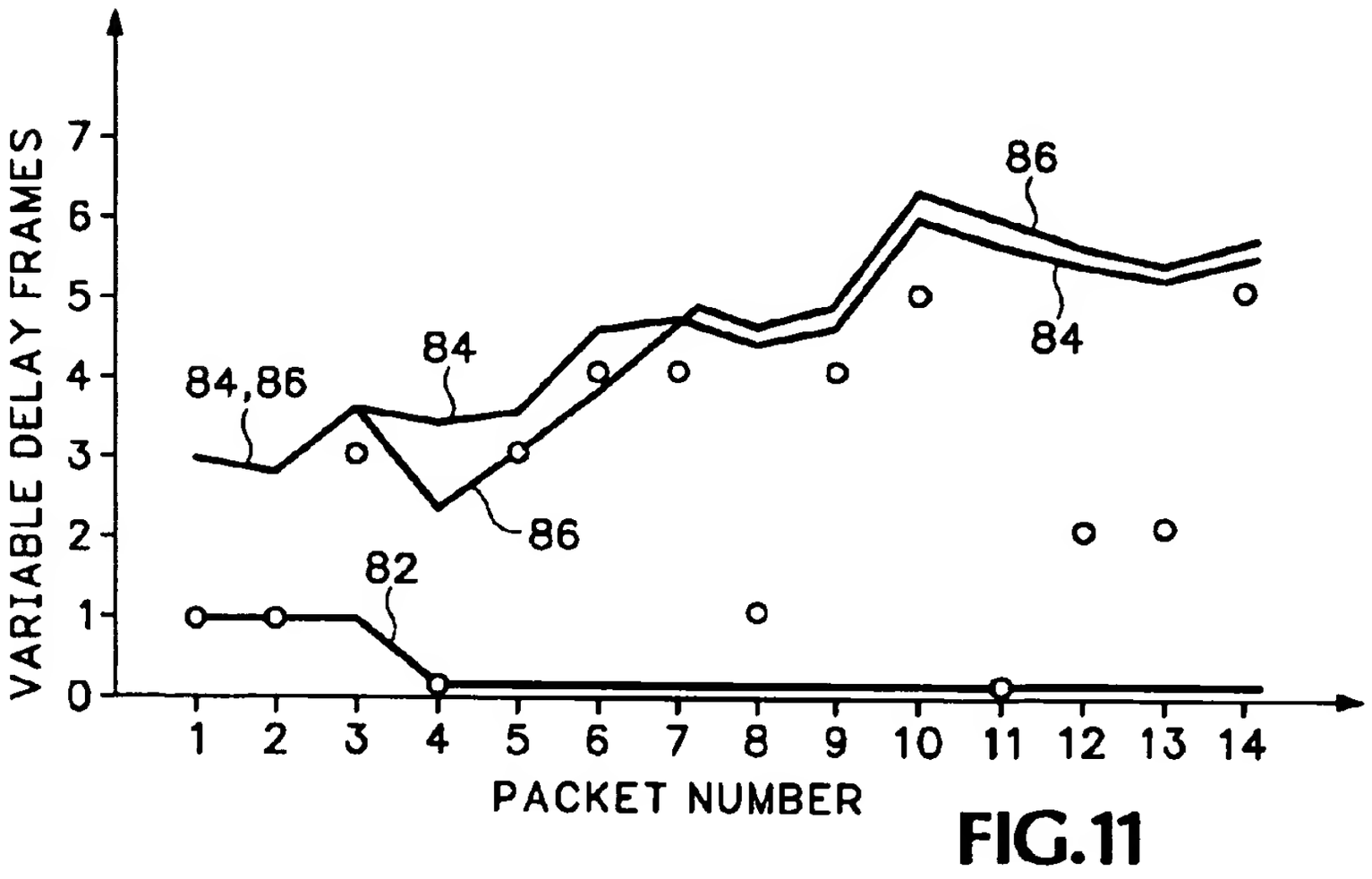


FIG.11

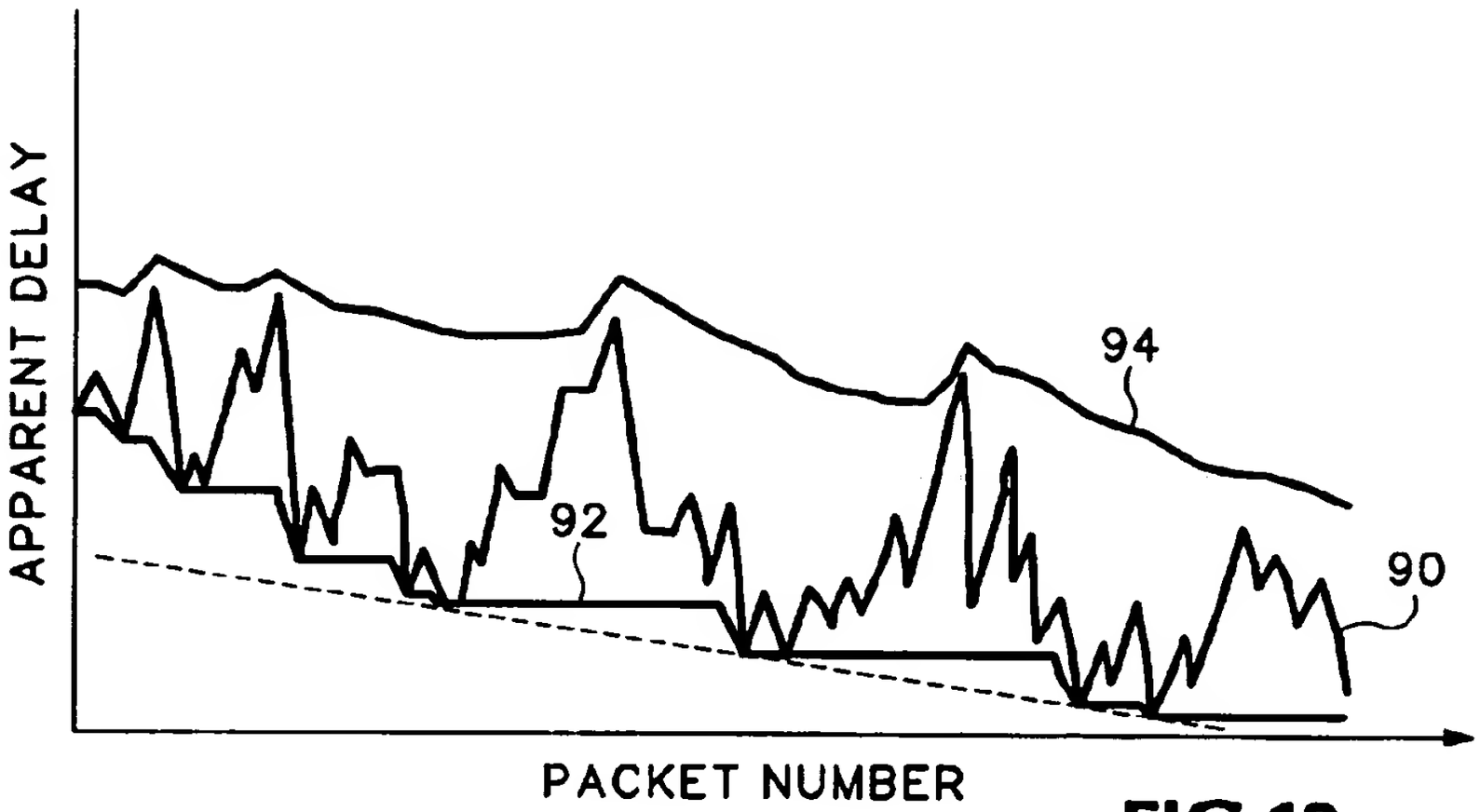


FIG.12

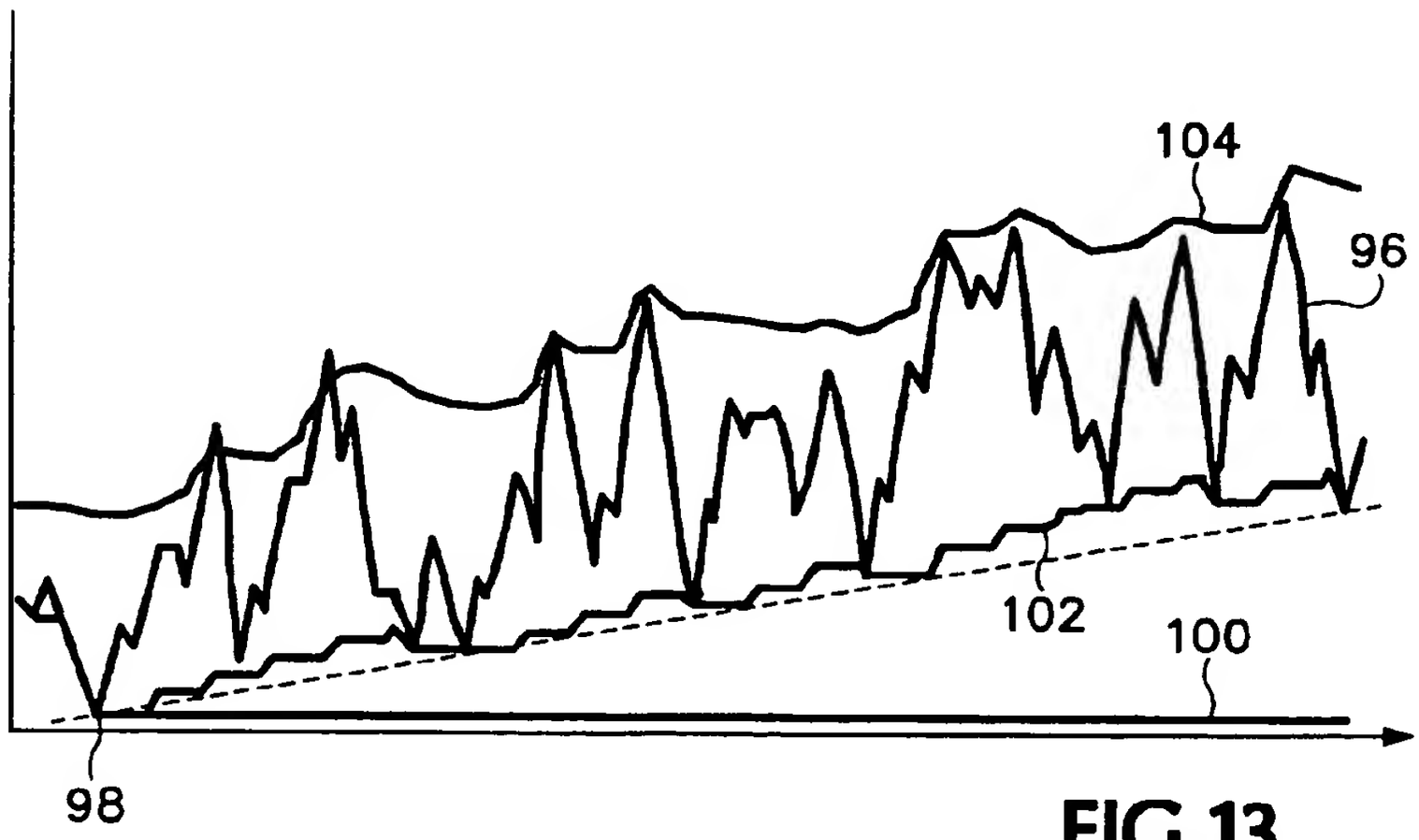


FIG.13

CLOCK SYNCHRONIZATION AND DYNAMIC JITTER MANAGEMENT FOR VOICE OVER IP AND REAL-TIME DATA

FIELD OF THE INVENTION

This invention pertains generally to methods and systems for communication of real-time audio, video, and data signals over a packet-switched data network, and more particularly to methods and systems for managing real-time data packet receipt and playout in the presence of variable packet delays.

BACKGROUND OF THE INVENTION

Most data networks are packet-switched. Data is communicated over a packet-switched network in small chunks, or "packets", which require no dedicated circuit. Each packet contains information that allows the data network to route it to the appropriate destination. Packets from many different senders travel sequentially over single connections between routing points, and packets from the same sender may travel different routes as network conditions change. Consequently, consecutive packets from a specific sender to a specific receiver may experience different delays as they travel different routes or experience different competing traffic loads along the network.

Researchers have sought ways to communicate real-time information over packet-switched data networks in order to take advantage of the time-varying nature and information redundancies found in most real-time data. For example, it is now possible to route voice telephone traffic over data networks through a technique commonly referred to as "Voice Over IP", or "VoIP" for short. VoIP can require significantly less average bandwidth than a traditional circuit-switched connection for several reasons. First, by detecting when voice activity is present, VoIP can choose to send little or no data when a speaker on one end of a conversation is silent, whereas a conventional, circuit-switched telephone connection continues to transmit during periods of silence. Second, the digital audio bitstream utilized by VoIP may be significantly compressed before transmission using a codec (compression/decompression) scheme. Using current technology, a telephone conversation that would require two 64 kbps (one each way) channels over a circuit-switched network may utilize a data rate of roughly 8 kbps with VoIP.

The variation in packet arrival rate, or "jitter", existing on most packet networks, presents challenges for real-time communication. To compensate for jitter, a real-time receiver must buffer packets for an amount of time sufficient to allow orderly, regular playout of the packets. Researchers have long recognized the need for an accurate method of receiver playout buffer length selection in real-time packet data communications such as VoIP. If the buffer delay is too short, "slower" packets will not arrive before their designated playout time and playout quality suffers. If the buffer delay is too long, it noticeably disrupts interactive communications. Selection of a near-optimal packet buffer delay for real-time communications requires accurate knowledge of actual packet delays.

Various protocols have been suggested for allowing receivers to obtain delay information. These include two described by W. Montgomery, "Techniques for Packet Voice Synchronization", IEEE J. on Selected Areas in Comm., vol. SAC-1, No. 6, pp. 1022–1028, Dec. 1983. One protocol uses an absolute clock reference by both a sender and a receiver. The sender timestamps each packet, and the receiver compares the timestamps on packets it receives to the absolute clock reference to determine delay. A second protocol would require that each packet switch along the network update a packet delay field to include the amount of time the packet was delayed by the switch. Since switches are the major source of variations in delay, the receiver can estimate delay by examining the delay field in received packets.

Unfortunately, neither of the protocols mentioned above are in widespread use today. Instead, most real-time packet data transmissions utilize the Real-time Transport Protocol (RTP). A sender using this protocol includes a packet timestamp generated from a local clock. The clock rate used to generate consecutive RTP timestamps is the clock rate of the data being transmitted—thus two consecutive packets should carry timestamps that differ by the number of data samples contained in the first of the two packets. Although RTP timestamps allow a receiver to reassemble samples in correct order, they contain no absolute delay information because the sender and receiver local clocks are not synchronized.

Despite the lack of absolute delay information in RTP headers, researchers have found ways to use adaptive, rather than fixed, buffer delays with RTP data streams. Although a fixed playout buffer delay can work in some circumstances (particularly with real-time communication over local area networks), adaptive playout buffer delay methods will generally perform better over a range of network conditions. An adaptive method attempts to minimize delay for current network conditions. Most techniques for adaptively adjusting buffer delay base their adjustments on statistics gleaned from RTP (or similar) timestamp histories. Four such techniques are discussed in R. Ramjee, et al., "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks" in Proceedings of the Conference on Computer Communications (IEEE Infocom), (Toronto, Canada), pp. 680–688, June 1994.

Each technique discussed in Ramjee et al. computes a delay estimate d_i and a delay variation v_i for each packet i. The basic adaptive algorithm is illustrated in FIG. 1. A packet i, containing a timestamp ts_i affixed to packet i by the sender, is received from packet-switched network **20** by receiver **16**. Summer **24** subtracts timestamp ts_i from a receive timestamp tr_i, taken from receiver clock reference **22**, to produce a difference sample n_i. With RTP, this difference will include an offset equal to the difference between the sender and receiver clock references. First-order filter **26** computes a mean delay estimate $\hat{d}_i$ from difference samples n_i. Summer **28** feeds the absolute value of the difference between $\hat{d}_i$ and n_i to a second filter **30**, which uses these samples to create a filtered estimate of the variation in delay $\hat{v}_i$. Multiplier **32** produces a multiple k of $\hat{v}_i$, which summer **34** adds to $\hat{d}_i$ and $\hat{ts}_i$ to create a playout time p_i for packet i.

Ramjee et al.'s other three discussed methods comprise various heuristic adaptations of the adaptive playout delay estimator of FIG. 1. One adaptation uses different time constants for filter **26**, depending on whether the latest measurement n_i will increase or decrease delay estimate $\hat{d}_i$. Another adaptation suspends delay estimate filtering temporarily if it detects a "spike" in the packet arrival rate. A fourth algorithm dispenses with filter **26** altogether, by examining all n_i computed for the last talkspurt received and setting $\hat{d}_i$ to the minimum of these values for the next talkspurt.

SUMMARY OF THE INVENTION

The present invention provides a packet-based real-time communication system utilizing an adaptive jitter manage-

ment system to reduce buffer latency while avoiding jitter underflow (jitter underflow occurs when the playout buffer runs out of data to playout). This system seeks to overcome several deficiencies in prior art adaptive systems, thereby providing increased performance over a wide variety of network conditions.

Variation in packet delay is not a stationary process. Despite this, most prior art algorithms attempt to estimate packet delay statistics with time-based estimates such as mean arrival time and variance from mean arrival time. Such algorithms tend to under perform at startup, as well as when packet delay statistical transitions occur. Furthermore, a "mean+rule-of-thumb times variance" playout estimate must be keyed to assumptions about the expected distribution of packet delays—because these assumptions will not always hold, the rule-of-thumb must be set conservatively. The prior art has attempted to cope with these deficiencies through a variety of heuristic adaptations, such as the statistical anomaly "spike" detector discussed in Ramjee et al. It is recognized herein that statistical estimation techniques are generally ill-suited for adaptive jitter control over a time-variant network.

The present invention avoids the statistical pitfalls of the prior art by basing playout buffer adjustments on the one stable statistic that exists in a packet-switched conference—fixed transmission delay. Instead of referencing statistical estimates to recent trends in the data, the present invention computes variable packet delays with reference to a minimum delay estimate valid for all received packets. The stability of the minimum delay statistic allows the present invention to accurately follow the jitter envelope of the variable packet delays and adjust playout time accordingly. A further benefit of the system is rapid convergence of the minimum delay statistic, which allows aggressive initial settings and good performance at connection startup.

In one aspect of the present invention, a packet-based real-time data receiver comprises a packet transmission fixed delay estimator and a packet transmission variable delay estimator. The fixed delay estimator keeps track of fixed delay (including offsets) over the duration of a conference connection. When a conference packet is received prior to a minimum arrival time predicted for that packet by the current fixed delay estimate, the fixed delay estimate is adjusted downwards (i.e., a packet with lower than the predicted fixed delay has been received, therefore the fixed delay estimate was too high). The packet transmission variable delay estimator calculates a variable delay for each received packet. A minimum arrival time based on the fixed delay estimate is subtracted from the packet's actual arrival time by the variable delay estimator.

The packet variable delays are preferably used by an adaptive playout delay estimator within the receiver. The adaptive playout delay estimator adapts packet playout delay in an attempt to reduce latency as much as possible without causing jitter underflow. In a preferred embodiment, this estimator performs a non-linear filter operation on the packet variable delays. The receiver may use the packet playout delay to control a playout buffer.

In another aspect of the present invention, a method of receiving and playing packetized real-time data is disclosed. When a real-time conference is established over a packet-switched network, a packet transmission fixed delay estimate and a playout delay estimate are initialized. A packet delay is calculated for each packet as it is received. The fixed delay estimate is adjusted downwards if the fixed delay estimate is greater than the packet delay for the current packet. A packet variable delay estimate is then obtained for the packet by subtracting the fixed delay estimate from the packet delay.

Preferably, the method further comprises non-linear adaptation of a playout delay estimate. In one embodiment, as packet variable delay estimates are calculated, they are filtered into the playout delay estimate using a non-linear gain filter. The gain of the filter is based on the ratio of the packet variable delay estimate to the playout delay estimate.

BRIEF DESCRIPTION OF THE DRAWING

The invention may be best understood by reading the disclosure with reference to the following figures:

FIG. 1, which shows a block diagram of a prior art adaptive playout delay system.

FIG. 2, which shows a breakdown of real-time data latency into its components on a timeline.

FIGS. 3 and 4, which show probability of packet arrival as a function of packet send time for two network delay distributions.

FIG. 5, which illustrates a packet-based real-time data receiver according to one embodiment of the present invention.

FIGS. 6 and 7, which illustrate two playout delay transfer functions useful with the present invention.

FIGS. 8–11, which compare a prior art adaptive playout delay method to a method according to one embodiment of the invention for two packet arrival sequences.

FIGS. 12 and 13, which illustrate performance of embodiments of the invention for skewed send and receive clock rates.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention generally applies to systems that receive real-time packet-switched data. Real-time data, as understood in the art, refers to data whose usefulness decays rapidly if delayed by more than a few seconds, such as interactive voice or video conferencing. One type of real-time data receiver that can employ the present invention is a computer connected to a packet network and either running VoIP software on its microprocessor, or having specialized VoIP hardware or firmware. The invention also applies to a data network telephony gateway. When a gateway operates as a receiver, it must buffer voice packets and output a continuous digital or analog stream onto a circuit-switched system. Other applicable systems include PBX equipment, packet network video or multimedia, and other real-time data delivery systems.

Packet Arrival Time Distributions

For real-time packet-switched data receivers, latency, i.e. the difference between packet send time and packet playout time, is of primary interest. With reference to FIG. 2, playout time t_p for a given packet is related to t_0, the time that the packet was constructed by the sender, by a concatenation of three delays. The first delay, d_f, represents the minimum travel time that a packet will incur in the network as it passes from sender to receiver. The second delay, d_v, represents the variable delay incurred by a packet in the network, e.g., due to competition with other network traffic. A packet is actually received at receive time $t_r=t_0+d_f+d_v$. The receiver places the packet in a buffer until the designated playout time t_p. The difference between playout time t_p and receive time t_r represents the buffer delay d_b set by the receiver for that packet—if d_b is set too low, t_r may exceed t_p for some

packets (i.e. late packets) and these packets will miss their playout time. Conversely, if d_b is set too high, packets will wait unnecessarily long for playout.

FIGS. **3** and **4** depict two probability distributions for packet arrival time, p_{pa}, as a function of t_0, over the duration of a conference. FIG. **3** shows p_{pa} as a Rayleigh distribution **60**, while FIG. **4** shows p_{pa} as a uniform distribution **62**. In both cases, the probability that a packet arrives prior to $t_f=t_0+d_f$ is zero. With a fixed playout time t_p, a few packets will arrive too late for playout if packets are distributed as shown by distribution **60**. For distribution **62**, all packets arrive well ahead of playout time t_p.

Most adaptive playout control systems attempt to estimate mean arrival time $\bar{t}_a$ and arrival time variance $\bar{v}_a$ for p_{pa}. These systems generally set $t_p=\bar{t}_a+k\bar{v}_a$, where k is a constant. As the system cannot know p_{pa}, it must set k conservatively (note that distributions **60** and **62**, as shown, have the same mean arrival time). And since p_{pa} is generally non-stationary, mean and variance may be difficult to estimate and track. Finally, variance itself contains some information of little value in setting buffer delay, i.e., information about the variation in packet arrival for packets that arrive before the mean arrival time (note that a minimally-delayed packet increases variance, thus increasing playout delay for such a system).

Fixed Delay and Variable Delay Estimation

The present invention abandons the concepts of mean arrival time and variance. Instead, an adaptive playout control system according to the invention estimates t_f, the fixed minimum arrival time for the conference. The fixed minimum arrival time is a stable statistic for all network packet arrival time distributions, both stationary and non-stationary. As will be shown, errors in the initial estimate of minimum arrival time can be corrected with no performance penalty. The preferred embodiments calculate packet jitter for each received packet as the difference between the minimum arrival time and the actual arrival time for that packet. Playout buffer delay is computed from packet jitter values.

FIG. **5** depicts an adaptive packet-based real-time data receiver according **30** to one embodiment of the present invention. Receiver **54** receives packets i from packet data network **20**, stores packet data in playout buffer **50**, and relays the send timestamp ts_i from packet i to the adaptive circuitry. Playout buffer control **48** computes a playout time p_i for each packet i, and releases packets to playout device **52** at their designated playout time.

Summer **40** computes a raw packet delay n_i for each packet i as the difference between the send timestamp ts_i and a receive timestamp tr_i. Generally, timestamps generated by the sending system and the receiving system are not synchronized. The present invention functions whether or not send and receive clocks are synchronized, although the remainder of the discussion assumes a lack of synchronization. Receive timestamp tr_i is computed from a receive clock. The receive clock utilizes a reference clock source related to the real-time data rate; preferably, the timestamp is supplied by playout buffer control **48**. Buffer control **48** preferably increments a timestamp counter each time a sample or frame of data is output to playout device **52**—this counter is a convenient reference source for tr_i.

Fixed delay estimator **42** uses raw packet delays n_i to compute a minimum packet delay estimate $\hat{dd}_f$. In its simplest form, fixed delay estimator **42** implements a floor function for all raw packet delays prior to and including raw delay for packet i, i.e., $\hat{d}_f=_{k=0 \text{ to } i}\lfloor n_k \rfloor$. This delay estimate is not a measure of absolute fixed delay, as it also contains the offset between the unsynchronized send and receive clocks (there is no mechanism to account for such a clock offset separately from a real fixed delay). Delay estimate $\hat{d}_f$ in this embodiment thus represents the minimum clock offset observed over the conference up to packet i.

Variable delay estimator calculates a packet jitter value j_i for each packet i. Packet jitter value j_i equals the estimated absolute variable delay for packet i. Packet jitter, or absolute variable delay, may be calculated by subtracting the clock offset and fixed delay (both contained in $\hat{d}_f$) from raw packet delay n_i. Packet jitter values are fed to adaptive playout delay estimator **46**, which in turn feeds playout delay values to playout buffer control **48**.

Packet-based real-time data receiver **54** may advantageously be implemented as a programmed microprocessor or signal processor. Although machine-level programming is processor-specific, the following pseudocode may be adapted to a specific processor for use in an adaptive playout control system of a real-time data receiver.

```
/* timestamp processing for each packet */
if (first packet)                                   /* initialization */
{
    /* set minimum packet delay to delay of first sample */
    fixed_delay = receive_clock - timestamp;
}
/* compute absolute variable delay for packet */
packet_jitter = receive_clock - timestamp - fixed_delay;
/* if packet delay is less than current minimum, adjust minimum */
if (packet_jitter < 0)
{
    fixed_delay = receive_clock - timestamp;
    packet_jitter = 0;
}
```

This code initializes the fixed delay estimate with a first timestamp difference. A packet jitter value is computed for each packet by subtracting the fixed delay from the timestamp difference for that sample. A negative packet jitter value indicates that the packet arrived before the minimum arrival time predicted by the current fixed delay estimate. In such a case, the fixed delay estimate is set to the timestamp difference of the new packet, and that packet's jitter is reset to zero.

Several safety measures may also be implemented in the above pseudocode. For instance, packets received out of sequence or otherwise suspect may be allowed to adjust packet_jitter in only small increments, e.g., one frame. Packets received very late may be marked so that they will not affect playout delay estimates at all. However, a long sequence (e.g., 8 packets) of consecutive very late packets may signify that an error has occurred that requires a reset of the adaptive playout system.

Playout Delay Estimation Using Variable Packet Delays

Jitter values as computed above are constrained to a time-varying envelope of arrival times bounded below by the fixed delay. The upper bound of this envelope must be set high enough to achieve acceptable late packet rates—for instance, for the ITU G.729 voice codec, voice quality degradation becomes noticeable if more than about 1.0% of transmitted voice packets miss their scheduled playout time. At the same time, talkspurts should generally be played out as soon as possible, dictating that the upper bound of the envelope adapt to recent packet jitter values.

A preferred embodiment of the invention includes a playout delay estimator—essentially, such an estimator adjusts an estimate of the upper bound of packet arrival times by comparing the current upper bound to measured

packet jitter values. A simple estimator operating on this principle adjusts delay by filtering a constant multiple k of observed jitter values. This delay estimate d_i, based on packet i and previous delay estimate d_{i-1}, may be expressed as

$$d_i=\alpha d_{i-1}+(1-\alpha)kj_i$$

This estimator functions acceptably when used with relatively time-stable packet arrival distributions having a low probability of $j_i>kd_{i-1}$. FIG. **6** illustrates an envelope estimator transfer function, having a nonlinear gain, that is particularly preferred for time-variant packet arrival distributions. No filter adjustment occurs with this filter for packet i if the ratio

$$\frac{j_i}{d_{i-1}}=\frac{1}{k}$$

As the ratio of packet jitter to delay estimate varies away from 1/k, the filter gain increases non-linearly, thus allowing the estimator to better track sudden variations in the arrival time upper bound.

In one embodiment, such a nonlinear estimator is approximated by applying different filters at different ranges, or zones, of the ratio of j_i to d_{i-1}. The following filter selection approximates nonlinear filtering with k=1.6 and avoids direct ratioing by division, instead comparing j_i to binary-shifted versions of d_{i-1}.

$$d_i=\begin{cases}\alpha_1 d_{i-1}+(1-\alpha_1)k\,j_i & j_i<0.25d_{i-1}\\ \alpha_2 d_{i-1}+(1-\alpha_2)k\,j_i & 0.25d_{i-1}\le j_i<0.50d_{i-1}\\ d_{i-1} & 0.50d_{i-1}\le j_i<0.75d_{i-1}\\ \alpha_3 d_{i-1} & 0.75d_{i-1}\le j_i<d_{i-1}\\ \alpha_4 d_{i-1} & d_{i-1}\le j_i\end{cases}$$

Gain factor settings used in one embodiment of the invention allows binary shifts and adds to be substituted for multiplies and divides; e.g., $\alpha_1=1-2^{-9}$,$\alpha_2=1-2^{-11}$, $\alpha_3=1+2^{-6}$, and $\alpha_4=1+2^{-2}$ for 20 msec packet sizes. This transfer function is illustrated in FIG. **7**. One characteristic of this setting is a quick envelope response to jitter values that approach or exceed delay estimate d_i (e.g., a 25% increase in d_i for jitter to delay estimate ratios greater than one). In contrast, the envelope responds relatively slowly to small jitter values. This behavior is desirable as it allows large jitter values a heavier weighting in the calculation of delay estimate d_i.

Playout Delay Estimation Examples

FIGS. **8**–**11** compare the response of a prior art mean/variance delay estimator to the response of a delay estimator according to the invention, for two sequences of variable packet delay. FIGS. **8** and **9** illustrate a first packet delay sequence (packet delays represented as circles). In these figures, the vertical baseline is the true fixed delay for the sequence.

FIG. **8** illustrates the response of a prior art receiver **16** as in FIG. **1** to the packet delay sequence. Curve **70** plots the mean estimate calculated by receiver **16**, and curves **72** and **73** show two playout delay estimates. Curve **72** uses a variance multiplier k=2, while **73** uses k=4 as discussed in Ramjee et al. Packet **1** of the sequence experiences a relatively high variable delay, resulting in a high initial estimate for mean **70**. As packet delays decrease towards the end of the sequence, playout delay estimates **72** and **73** remain high. This occurs not only because of the high initial mean estimate, but because the low-delayed packets (i.e. packets **4**, **7**, **11**, **12**) actually increase playout delay estimates **72** and **73** because they vary from the mean by a relatively large (although negative) amount. As a result, playout of the latter portion of the sequence may be delayed 2 to 5 frames longer than actually required for the sequence.

FIG. **9** shows the same packet variable delay sequence, along with a fixed delay estimate **74** and two playout delay estimates **76**, **77** according to embodiments of the present invention. Like mean estimate **70** above, fixed delay estimate **74** starts off badly in error because of the high delay of packet **1**. As each packet with a smaller delay than previously observed packets arrives (i.e. packets **2**, **4**, **7**), fixed delay estimate **74** tracks downward towards the true fixed delay. From packet **7** on, estimate **74** represents the true fixed delay of the connection.

Playout delay **76** follows the 5-region non-linear gain jitter filter methodology set out in FIG. **7** and in the section above for playout delay estimate d_i. The embodiment represented by delay **76** uses compensation to avoid direct mirroring of changes in fixed delay estimate **74** in playout delay estimate **76**. For instance, at packet two fixed delay estimate adjusts downwards two frames. Delay estimate d_i is adjusted upwards two frames at this point in compensation, such that playout delay **76** does not track fixed delay estimate **74** directly. Playout delay **76** accurately mirrors trends in packet delay over the sequence, while providing a one to two frame cushion.

Curve **77** represents playout delay calculated using a second embodiment of the invention. This embodiment differs from the embodiment producing delay **76** in that it does not compensate d_i for downward shifts in fixed delay **74**. Thus at packet **2**, playout delay **77** tracks the two-frame adjustment in fixed delay **74**, placing it lower than the actual delay of packet **3**. This causes the delay estimator to sharply increase d_i at packet **3**, although playout delay **77** drops again at packet **4** due to another adjustment in fixed delay **74**. Once fixed delay **74** stabilizes, curve **77** should begin to converge with curve **76**.

FIGS. **10** and **11** illustrate a second packet arrival sequence. FIG. **10** illustrates performance for prior art adaptive delay estimator **16**. Packet **1** experiences a relatively low delay, forcing a low initial mean estimate **78**. Other packets with low delay (packets **2**, **4**, **8**, **11**, **12**, **13**) negatively affect growth of playout delay **80** because of their low variance. Consequently, packet **3** arrives at the current playout estimate, and packets **6**, **7**, **9**, **10**, and **14** arrive too late for their estimated playout time with k=2 (curve **80**). Playout delay estimate **81**, with k=4, appears adequate, although this appearance is largely due to the low mean estimate.

FIG. **11** shows the same packet arrival sequence as FIG. **10**, this time using fixed delay adjustment-compensating (curve **84**) and non-compensating (curve **86**) embodiments as described in the description accompanying FIG. **9**. Fixed delay estimate **82** adjusts once, at packet **4**, where the minimum clock offset observed over the packet sequence occurs. Playout delay estimates **84** and **86** adjust rapidly to envelop the numerous long-delay samples in this sequence. After packet **4**, playout delay estimates **84** and **86** begin to converge.

FIGS. **8** through **11** illustrate different startup scenarios that an adaptive playout delay estimator may encounter. But such scenarios also represent statistical shifts in the packet arrival time distribution that may occur mid-conference. The minimum delay estimate of the invention provides a solid

reference during these shifts from which playout delay may be adjusted. As a result, the present invention rapidly detects and adjusts to increasing packet delays. Generally, this allows the present invention to maintain a more aggressive playout schedule than prior art systems.

Although receiver **54** preferably adjusts playout delay with every incoming data packet, the estimate preferably does not affect playout from buffer **50** (FIG. **5**) at every frame. Playout buffer control **48** utilizes the output of envelope estimator **46** to adjust delay only at the beginning of each talkspurt. Effectively, playout delay is modulated by shrinking or stretching the amount of time between consecutive talkspurts.

Compensating for Statistical Shifts in Fixed Delay

According to the present invention, a real-time packet receiver bases buffer length and playout delay on a fixed delay estimate. Problems may arise if this fixed delay is not truly "fixed" over the duration of a conference. The most common example of this is where the send clock and receive clock operate at slightly different rates, resulting in a constant bias rate in the computed packet timestamp differences. Another example of a shift in fixed delay involves the loss of a network path, forcing all packets to take a longer route. The present invention automatically corrects for negative bias rates and shifts (i.e., faster minimum packet arrivals), and with a slight modification, can correct for positive fixed delay bias rates and shifts also.

FIG. **12** illustrates a negatively rate-biased packet arrival sequence **90**. Fixed delay estimator **42** automatically tracks negative biases, which resemble "better" estimates of minimum delay. Fixed delay estimate **92** stairsteps downward as new samples with smaller clock offsets are received. Playout delay **94** may be configured to stairstep downwards with fixed delay estimate **92**. Optionally, and as shown, playout delay **94** does not automatically stairstep downwards with every step of **92**, but relies on its envelope-following characteristics to track the negative rate-bias in packet arrival sequence **90**.

FIG. **13** illustrates a positively rate-biased packet arrival sequence **96**. The minimum observed packet arrival occurs at point **98** in sequence **96**. Using the basic fixed delay estimator of the present invention, fixed delay would remain at the value observed at point **98**, as shown by curve **100**, for the remainder of the conference. Over time, a large offset may develop between the true and the estimated fixed delay, resulting in unnecessary playout delay, suboptimal variable delay estimation, and possible eventual playout buffer exhaustion (depending on how the buffer is implemented).

To combat the positive rate-bias problem, it is preferred that a small positive rate bias be introduced artificially into the fixed delay estimate. One method of accomplishing an artificial bias is to count packets since the last downward update to the fixed delay estimate. If the counter reaches a set target value, the fixed delay estimate is increased, e.g., by one frame. If the data has no actual positive rate-bias, a subsequent low-delay packet should quickly re-adjust the fixed delay estimate back down and reset the bias counter. Fixed delay estimate **102** illustrates how the artificial rate bias allows estimator **42** to track a positive rate bias in sequence **96**.

In practice, most biases will be unnoticeable over the length of a conference. A low artificial bias rate, e.g., equivalent to one sample/packet, will generally be more than sufficient. If new low-delay packets are not observed after adjustment of the fixed delay upwards, the artificial bias rate may optionally be increased gradually until a new low-delay packet is found. One method of increasing bias rate is to reduce the set target value the counter must reach each time an artificial up-adjustment with no preceding down-adjustment is made.

The invention has been described herein with reference to several illustrative embodiments. Other modifications to the disclosed embodiments will be obvious to those of ordinary skill in the art upon reading this disclosure, and are intended to fall within the scope of the invention as claimed. For example, many possible variations exist for an envelope estimator—the present invention teaches that such an estimator have the capability to decrease playout time in response to observed jitter values much lower than the current playout delay, and relatively rapidly increase playout time in response to observed jitter values of roughly the same magnitude or higher than the current playout delay. Likewise, other methods of implementing positive-rate-bias detection and compensation for fixed delay estimation will be immediately obvious to one of ordinary skill upon reading this disclosure. The particular playout buffer implementation is not critical to the present invention. Numerical values disclosed herein are tuning parameters that may be adjusted for a given application using the principles taught in this disclosure.

What is claimed is:

1. A packet-based real-time data receiver comprising:

a packet transmission fixed delay estimator, said fixed delay estimator keeping a fixed delay estimate over the duration of a conference connection using said receiver, and adjusting said fixed delay estimate downwards during said conference in response to the arrival of a conference packet prior to a minimum arrival time predicted for that packet by said fixed delay estimate; and

a packet transmission variable delay estimator, said variable delay estimator calculating a variable packet delay for each conference packet by subtracting a predicted minimum arrival time for each packet, based on said fixed delay estimate, from the actual arrival time of each packet.

2. The data receiver of claim **1**, further comprising an adaptive playout delay estimator that adapts packet playout delay for said receiver using variable packet delays from said packet transmission variable delay estimator.

3. The data receiver of claim **2**, wherein said adaptive playout delay estimator comprises a non-linear packet variable delay filter.

4. A packet-based real-time data receiver comprising:

a playout buffer for queuing packets from a received data stream for playout, said packets in said received data stream each containing a packet timestamp generated by a remote system send clock operating at a send clock rate;

a local timestamp generator operating at approximately said send clock rate;

a packet transmission fixed delay estimator, said fixed delay estimator comparing the packet timestamp from each received packet to a receive timestamp taken from said local timestamp generator at approximately the arrival time of said received packet, and adjusting a fixed delay estimate downwards in response to the arrival of packets prior to an arrival time predicted by said fixed delay estimate;

a packet transmission variable delay estimator, said variable delay estimator calculating a variable delay for each packet by subtracting said fixed delay estimate from the difference between the receive timestamp and the packet timestamp; and

a playout delay estimator that non-linearly adapts a playout delay estimate for received packets based on the relative magnitude of the playout delay estimate as compared to a variable delay calculated by said packet transmission variable delay estimator.

5. The data receiver of claim 4, wherein said local timestamp generator is synchronized with said remote system send clock.

6. The data receiver of claim 4, wherein said packet transmission fixed delay estimator, packet transmission variable delay estimator, and playout delay estimator comprise a programmed microprocessor.

7. A method of receiving and playing packetized real-time data, said method comprising the steps of:

establishing a real-time conference over a packet-switched network;

for a first real-time data packet received during said conference, initializing a packet transmission fixed delay estimate and a playout delay estimate for the conference; and

for each additional real-time data packet received during said conference,

calculating a packet delay estimate,

adjusting said fixed delay estimate downwards if said fixed delay estimate is greater than said packet delay estimate, and

subtracting said fixed delay estimate from said packet delay estimate, thereby obtaining a packet variable delay estimate.

8. The method of claim 7, wherein said step of initializing a packet transmission fixed delay estimate consists of setting said fixed delay estimate to equal the clock offset between a local clock reference and a timestamp affixed to said first data packet by its sender.

9. The method of claim 7, wherein said adjusting said fixed delay estimate downwards step comprises, for a data packet triggering such adjustment, resetting said fixed delay estimate to equal the clock offset between a local clock reference and a timestamp affixed to that data packet by its sender.

10. The method of claim 7, further comprising as a part of said adjusting said fixed delay estimate downwards step, adjusting said playout delay estimate upwards by an equivalent amount.

11. The method of claim 7, further comprising the step of introducing an artificial positive rate bias in said fixed delay estimate.

12. The method of claim 7, further comprising the step of adapting said playout delay estimate throughout the duration of said conference to approximately maintain a preset ratio between said playout delay estimate and said packet variable delay estimates calculated for said real-time data packets.

13. The method of claim 12, wherein said adapting said playout delay estimate comprises applying a correction formula to said playout delay estimate for each packet variable delay estimate, with a correction formula gain determined by the ratio of that packet variable delay estimate to the playout delay estimate.

14. The method of claim 13, wherein said gain is highest for packet variable delay estimates greater than the playout delay estimate and less than a preset maximum variable delay.

15. The method of claim 13, comprising applying said correction formula by mapping said ratio into one of a plurality of ratio zones, each of said zones having a zone-specific gain formula.

16. The method of claim 7, further comprising the step of fixing the playout time for data spurts during said conference using said playout delay estimate.

17. The method of claim 7, wherein said step of adjusting said fixed delay estimate downwards comprises resetting said fixed delay estimate to said packet delay estimate.

18. A method of receiving and playing packetized real-time data, said method comprising the steps of:

establishing a real-time conference over a packet-switched network;

for a first real-time data packet received during said conference, initializing a packet transmission fixed delay estimate and a playout delay estimate for the conference;

introducing an artificial positive rate bias in said fixed delay estimate;

for each additional real-time data packet received during said conference,

calculating a packet delay estimate,

adjusting said fixed delay estimate downwards if said fixed delay estimate is greater than said packet delay estimate, and

subtracting said fixed delay estimate from said packet delay estimate, thereby obtaining a packet variable delay estimate;

adapting said playout delay estimate throughout the duration of said conference to approximately maintain a preset ratio between said playout delay estimate and said packet variable delay estimates calculated for said additional data packets; and

periodically adjusting playout time for data packets received during said conference to include a buffer time, measured with reference to the fixed delay estimate, corresponding to said playout delay estimate.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,259,677 B1
DATED : July 10, 2001
INVENTOR(S) : Jain

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 2,
Line 51, "$\hat{i}\ \hat{s}_i$" should read -- ts_i --

Column 4,
Line 63, "$t_r = t_o + d_f = d_v$" should read -- $t_r = t_o + d_f + d_v$ --

Column 5,
Line 41, "receiver according **30** to" should read -- receiver **54** according to --
Line 63, " $d\hat{d}_f$ " should read -- $\hat{d}_f$ --

Column 6,
Line 6, "estimator calculates" should read -- estimator 44 calculates --

Signed and Sealed this

Seventh Day of May, 2002

Attest:

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

Attesting Officer